NUREG/CR-2364 Vol. V
ANL-81-59 Vol. V

NUREG/CR-2364 Vol. V
ANL-81-59 Vol. V

PROJECTION MODELS FOR HEALTH EFFECTS ASSESSMENT IN POPULATIONS EXPOSED TO RADIOACTIVE AND NONRADIOACTIVE POLLUTANTS

Volume V

SPAHR Programmer's Guide

by

James J. Collins and Robert T. Lundy

ARGONNE NATIONAL LABORATORY

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Prepared for the Office of Nuclear Regulatory Research
U. S. NUCLEAR REGULATORY COMMISSION
under Interagency Agreement DOE 40-550-75

NUREG/CR-2364 Vol. V
ANL-81-59 Vol. V

(Distribution
Code: RH)

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

PROJECTION MODELS FOR HEALTH EFFECTS ASSESSMENT IN POPULATIONS EXPOSED TO RADIOACTIVE AND NONRADIOACTIVE POLLUTANTS

Volume V

SPAHR Programmer's Guide

by

James J. Collins and Robert T. Lundy

Division of Biological and Medical Research

September 1982

Prepared for the Health Effects Branch
Division of Health, Siting and Waste Management
Office of Nuclear Regulatory Research
U. S. Nuclear Regulatory Commission
Washington, D. C. 20555
Under Interagency Agreement DOE 40-550-75

NRC FIN No. A2059

This is Volume V of a five volume series entitled Projection Models for Health Effects Assessment in Populations Exposed to Radioactive and Nonradioactive Pollutants, NUREG/CR-2364, ANL-81-59. The series presents version 4.1 of the Simulation Package for the Analysis of Health Risk (SPAHR) computer package and model. The complete series of SPAHR documentation is contained in the following five volumes:

Volume I	Introduction to the SPAHR Demographic Model for Health Risk J. J. Collins, R. T. Lundy, D. Grahn, and M. E. Ginevan
Volume II	SPAHR Introductory Guide J. J. Collins and R. T. Lundy
Volume III	SPAHR Interactive Package Guide J. J. Collins
Volume IV	SPAHR User's Guide J. J. Collins and R. T. Lundy
Volume V	SPAHR Programmer's Guide J. J. Collins and R. T. Lundy

PROJECTION MODELS FOR HEALTH EFFECTS ASSESSMENT IN POPULATIONS EXPOSED TO RADIOACTIVE AND NONRADIOACTIVE POLLUTANTS

ABSTRACT

The Simulation Package for the Analysis of Health Risk (SPAHR) is a computer software package based upon a demographic model for health risk projections. The model extends several health risk projection models by making realistic assumptions about the population at risk, and thus represents a distinct improvement over previous models. Complete documentation for use of SPAHR is contained in this five-volume publication. The demographic model in SPAHR estimates population response to environmental toxic exposures. Latency of response, changing dose level over time, competing risks from other causes of death, and population structure can be incorporated into SPAHR to project health risks. Risks are measured by morbid years, number of deaths, and loss of life expectancy. Comparisons of estimates of excess deaths demonstrate that previous health risk projection models may have underestimated excess deaths by a factor of from 2 to 10, depending on the pollutant and the exposure scenario. The software supporting the use of the demographic model is designed to be user oriented. Complex risk projections are made by responding to a series of prompts generated by the package. The flexibility and ease of use of SPAHR make it an important contribution to existing models and software packages.

FIN #	Title
A2059	Projection models for health effects assessment in populations exposed to radioactive and nonradioactive pollutants

TABLE OF CONTENTS, VOLUME V

TABLE OF CONTENTS

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Prediction of the health consequences to the general population of exposure to airborne and waterborne pollutants is becoming an important feature of environmental impact analyses. Such prediction requires not only knowledge of the dose term and the dose-response function, but also a model for projecting the health risk to some future population. Health risk projections entail considerable uncertainty about the measurement of the dosage that individuals receive and about the magnitude and nature of the biological response at a given population exposure. The uncertainties regarding the individual dose and the dose-response function have received much attention, but the uncertainty associated with the health risk projection model itself has not been fully addressed.

The purpose of this publication is threefold. First, the uncertainties in various health risk projection models will be addressed, and the assumptions inherent in each model will be stated explicitly. Second, a new model that is an extension of earlier models will be introduced. It is argued that this new model, referred to as the demographic model, is superior to previous models because it makes fewer assumptions about the population at risk and the potential of the population to change over time. Third, a computer package referred to as the Simulation Package for Analysis of Health Risk (SPAHR) is presented which facilitates the application of this model for various pollutants and populations at risk.

The core of any risk assessment scheme is the exposure-response model. This is the quantitative relationship between the level of exposure to the hazard of interest and the deleterious effects resulting from that hazard. If the population exposed to the hazard is homogeneous with respect to its likelihood of suffering ill effects from the exposure, estimation of effects is straightforward; we need know only the total number of persons exposed to estimate the effects. However, if the population is heterogeneous (i.e., different persons have differing risks of suffering health effects from exposure to the hazard), then a reasonable assessment of population risk depends upon the distribution of persons by level of risk.

Research indicates that risk levels are often related to the age and sex characteristics of the exposed population. This is true for both radiation and air pollution exposures. When the risk level is a predictable function of age and sex or some other traceable component of the demographic structure of the population, the estimation of projected health effects becomes less straightforward. If one adds to this complexity the long latency periods between exposure and response, the competing risks from other causes of mortality, and the changing demographic structure of the population over time, the projection of health effects becomes even more complex.

Evaluation of the health consequences for populations exposed to pollutants has become an important issue because of the increasing number of known

or suspected carcinogens in the environment. To date, three projection methods have been used in health risk assessments: the single coefficient model, the multi-coefficient model, and the life table model. Each has its own short-comings, as discussed in Volume I, Chapter 2. This document presents a fourth model that is more useful and realistic than the previous models because it incorporates age, fertility, and mortality structure, and can follow populations through time under changing levels of mortality, fertility, and pollution exposure. This model is referred to as the demographic model.

A sensitivity analysis of the demographic model indicates that population structure alone for a 100-year exposure to 1 rem may introduce more than a factor of 10 variation in the number of excess deaths. This finding substantiates the premise that the population structure may be more important in a health risk projection than the uncertainty inherent in the dose-response functions.

A comparison of the demographic model with the single coefficient model, the most widely used in health risk projections, is presented in Volume I, Chapter 7. It is concluded that the single coefficient model, even in a short-term projection, may seriously underestimate excess deaths since it is unable to accumulate exposure. For instance, comparison of the single coefficient model with the demographic model for continuous exposure to 0.87 ppb of benzene for 50 years yields widely different estimates of excess mortality. The single coefficient model estimates 2,250 deaths, while the demographic model estimates values from 6,386 to 17,568. In the years 2015-2020, the excess leukemia deaths projected by the demographic model are ten times as large as those of the single coefficient model.

The demographic model is also compared with the life table model used in the 1980 BEIR report to estimate excess cancer deaths from exposure to ionizing radiation. The life table model correctly estimates the increased individual probability of death associated with a given radiation scenario. However, the life table model yields misleading results in the estimation of excess deaths for a specific population. The results presented in the 1980 BEIR report underestimate excess deaths by 50% in some instances. For example, using the linear-quadratic, absolute risk model for a continuous exposure of 1 rad per year for 70 years, the life table model estimates 2459 excess male deaths per million while the demographic model estimates 3769 excess male deaths per million.

This document is divided into five volumes:

I. Introduction to the SPAHR Demographic Model for Health Risk

II. SPAHR Introductory Guide

III. SPAHR Interactive Package Guide

IV. SPAHR User's Guide

V. SPAHR Programmer's Guide

The first volume presents the theory behind the SPAHR health risk projection model and several applications of the model to actual pollution episodes. The elements required for an effective health risk projection model are specified, and the models that have been used to date in health risk projections are outlined. These are compared with the demographic model, whose formulation is described in detail. Examples of the application of air pollution and radiation dose-response functions are included in order to demonstrate the estimation of future mortality and morbidity levels and the range of variation in excess deaths that occurs when population structure is changed. Volumes II through V provide the potential user with detailed guidance and appropriate examples to aid in the interpretation of numerical demographic output from the application of the model to realistic circumstances.

1.0 THE SPAHR PROGRAMMING ENVIRONMENT

The SPAHR system contains an executive command processor (subroutine SPAHR) whose major function is to read the first character string in each command statement and go to the appropriate command processor. A command processor, in turn, is a subroutine whose function is to interpret the remainder of the command statement and take whatever action is appropriate. SPAHR command processors are of two kinds: language processor routines that control specialized functions of the language processor, such as loop control, conditional execution of other commands, system variable definition, and other Input/Output (I/O) activities; and working routines that perform demographic calculations. SPAHR is designed so that a working routine is tied to the SPAHR system only at a few easily isolated points. It is thus possible to develop a working command processor as an independent program.

From the point of view of the programmer, then, SPAHR is a set of nearly independent subroutines or sets of subroutines tied together by a set of common blocks and a moderately sophisticated language processor. The language processor consists of a set of very general routines designed to simplify the use of the specialized demographic routines by allowing the use of free-format, English-like statements rather than a rigidly formatted set of input cards to control program execution. This enables the user to concentrate efforts on the program without undue concern about placement of a particular number or symbol and makes it easier to run the program in an interactive, time-shared environment. Furthermore, it relieves most users of the necessity for knowledge of FORTRAN programming and permits a low level of interaction with the operating system of the computer. However, for users who wish to extend or modify the capabilities of the SPAHR system, the language processor forces a number of conventions and restrictions.

2.0 LANGUAGE PROCESSOR INTERFACING ROUTINES

2.1 General Overview

A command cannot be used by a computer until it has been transferred to the central memory of the computer and reorganized so that each item of information contained in the command has been used to set a particular bit at a particular location on or off. The computer can then use the command for calculations or decisions. The tasks of transfer and reorganization are accomplished in SPAHR by a set of subroutines and functions that will be introduced below in a narrative describing each step in the interpretation of a command line. Most of the steps and routines included in this description are performed automatically through calls to the higher-level routines that will be described later. However, this discussion will help the user of the higher-level routines to understand them at a primitive level.

In order to interpret a series of characters from an input record, the input must first be transferred to the computer's memory. This transfer is done by routine GETCRD, which moves the next record it finds into a buffer named IC. A pointer named I1 points to the next character in the IC array to be processed. Whenever one of the processing routines is called, it processes the characters in the IC array starting with the one pointed to by I1. When the routine is finished, it updates I1 to point to the next character after the last one used. (Note below an exception to this rule for DELIM.)

The characters in the record must be identified and processed. The type of processing to be done depends on the type of characters, so a routine named AHEAD is called that returns an index telling whether the next nonblank character in the record is a letter, number, or special character. Then the program may call SCAN, which copies the character string into an 8-character variable, or RDNUM, which translates numbers into internal representation. When this is done, the program may call routine DELIM, which returns an index informing the program about the way the string was terminated. This can be exceedingly important, because an equal sign (=), for example, would indicate that whatever follows is important in the interpretation of what was just read in, while a semicolon (;) would imply under most circumstances the end of processing for the current command.

Interpreting a SPAHR command could be a long and tedious process. A set of higher-level routines is therefore provided that performs most of the repetitive tasks described above automatically. The routines GETPAR, RDLA, RDLA2, and RDRV, given an array of keyword names and arrays of space in which to store the associated data, will go through the laborious process of calling AHEAD, SCAN, RDNUM, DELIM, and other utility routines and will return the appropriate data. Of the primitive routines described above, only SCAN is likely to be needed by the user, and then only if the command processor will be reading in arrays or subcommands.

Chapter 2 describes each of the primitive and high-level routines that the user might employ in writing a SPAHR command processor. The language processor contains many other routines that should never be called from a command module. In the discussion that follows, variable names that are not formal parameters in the subroutines described are members of common block, CCARDC, unless otherwise specified. All routines are SUBROUTINE subprograms unless identified in some other way.

2.2 Language Notes

The SPAHR program is written exclusively in IBM G-level FORTRAN. This language is fairly old and contains some deficiencies, most notably the lack of a CHARACTER variable type and the absence of any form of internal format conversion such as ENCODE and DECODE.

The absence of CHARACTER variables is treated by using double precision (REAL*8) variables to store strings as long as eight characters. These strings may in turn be manipulated by denoting their equivalence to single-byte logical variables (type LOGICAL*1). Relational comparisons between single characters are performed by copying the single byte into half of a halfword integer (once again by using equivalenced variables) and performing the relational comparison on it.

To handle those few cases in which format conversion is desirable, some special routines to convert character strings into numbers and vice versa were written in FORTRAN. These routines are not discussed in this document, because it is anticipated that they will be replaced as compilers implement the 1977 FORTRAN standard when it becomes available.

2.3 High-Level Routines

Six high-level routines are in the language processor, GETPAR, RDLA, RDLA2, DFILE, TITL, and DEMERV. Each of these routines is discussed below.

2.3.1 GETPAR (KW,NPAR,IP,RP)

GETPAR reads in the values of keyword parameters. The next string found is compared against the NPAR members of KW, which contains 8-character keyword names. If a match is found, and the delimiter is an equal sign (=), RVALUE is called to place the numeric value following the = in the corresponding member of RP or IP. If the first letter of the keyword is I through N, an integer will be placed in IP. Otherwise, a floating point quantity will be placed in RP. If the delimiter is anything other than =, a -1 is placed in IP. The process will be repeated until a delimiter $$, $, or ; is found.

2.3.2 RDLA (V,LV,L1,L2,L1MAX,IFOUND)

RDLA reads a labeled two-dimensional array V(L1,L2), where the column labels are contained in the array of eight-character names LV. L1MAX returns the greatest length of any column read, and the length of each column entered is returned in the corresponding member of array IFOUND. A 0 is returned in the corresponding IFOUND member if the column was not read in. Each column is identified by using calls to SCAN and INDEX8; when identified, RDRV is called to read in the set of numbers. Subroutine ARITH is called to permit equation processing as an alternative method of initializing array columns if the last qualifier is delimited by = instead of a blank. ARITH is called whenever a two-dimensional, labeled array is to be read from the command file or a free-format data file. Labeled arrays are discussed in Chapter 3.4 of the User's Guide for SPAHR (Volume IV in this series). A two-dimensional array in SPAHR has only one level of qualifier names following the array name.

V	is the data array. It is a single-precision, floating-point array declared in two dimensions V(L1,L2) in the calling program.
LV	is an array of eight-character qualifier names. It is declared as a double precision (REAL*8) array with L2 members.
L1	is the maximum number of numbers that may be read into each labeled section of the array.
L2	is the maximum number of labeled sections in the array, and hence the maximum number of labels as well.
L1MAX	is the maximum number of numbers following any label.
IFOUND	is an integer array of dimension L2 whose members give the number of numbers following each of the corresponding labels in the LV array. If the corresponding label was not mentioned in the input, then the IFOUND entry returns a zero.

2.3.3 RDLA2 (V,LV,LV2,L1,L2,L3,L1MAX,IFOUND)

RDLA2 reads a labeled three-dimensional array. The user calls to SCAN and INDEX8 to identify the groups defined by the names in the LV array, and then sets up calls to RDLA based on the LV2 array names. ARITH may be invoked at the LV name level or the LV2 name level directly from RDLA.

2.3.4 DFILE (FILE,IFILE,IFLAG,DEF,IDEF)

The DFILE routine simplifies file definition and use in a SPAHR command module. It should be invoked following a GETPAR call in which one of the parameters was a file unit number. The corresponding member of the parameter array that was used as an argument to GETPAR should be used as the parameter FILE in the call to DFILE.

The arguments are treated as follows:

FILE	is a control parameter read in using GETPAR. If it is greater than or equal to 1.0, its integer value is returned in IFILE unchanged, and IFLAG is set to -1. If FILE is equal to 0.0, this indicates that the user does not wish to print any of the data intended for file unit IFILE in the calling routine, so IFLAG is set to 0 and IFILE is set to the value of DEF. If DEF was entered as 0.0, IFILE will return with the value of IDEF. If FILE has a negative value, DFILE assume that the printing is to be done on the default file, and therefore sets IFLAG to -1 and IFILE to DEF or IDEF as indicated above. FILE will have a negative value if the name of the parameter was encountered during the GETPAR call and was not followed by an = sign, causing GETPAR to set a default value of -1.
IFILE	is the positive integer value returned for the file unit number.
IFLAG	is the use/nonuse index for the file unit. If it is returned as -1, the calling routine assumes that I/O operations coded for unit IFILE are to be performed. An IFLAG value of zero indicates that the I/O operations are to skipped.
DEF	is the floating point default value for IFILE. It will be passed from the calling program as a value from a common block defining a global default for similar files. Therefore, it can have a value of zero indicating that the default is no input/output for this file.
IDEF	is the integer default value for IFILE. It must always be set on entry as a positive integer between one and the maximum possible value for a FORTRAN file unit.

2.3.5 TITL (IFILE,N1,IDENT,N2,NAME)

This routine produces a page heading on the print file with FORTRAN file unit number IFILE. It increments the page counter IPAGE (IFILE), prints out a numbered page title, and clears the line number counter LINES (IFILE) to zero. The page title consists of a line at the very top of the page containing the contents of array TITLJ in /TITLEC/ and the page number, followed by a blank line, followed by the two strings IDENT and NAME of N1 and N2 characters, respectively. The TITLJ array is processed through routine CNVS before printing to remove excess blanks and to expand the values of any SPAHR variables preceded by ampersands that may have been included when the TITLJ array was initialized. TITLJ is usually initialized by the TITLE command in SPAHR.

2.3.6 DEMERV (IERR,LPGM,CLUE,IN1,IN2)

This routine is the general error trap for SPAHR command modules. When called, it prints out a message determined by the value of IERR. Then, if

SPAHR is running in interactive mode, DEMERV returns to the calling program. If in batch mode, however, DEMERV issues a program stop. To cover interactive use, the calling routine should then arrange for an expeditious return to the routine that called the command processor (subroutine DEMPAK), so that further work can be done. IERR, IN1, and IN2 are single-precision integers. LPGM and CLUE are eight-character strings.

The messages associated with each value of IERR are as follows:

0	No message printed.
1	(LPGM) HAS NO COMMAND (CLUE).
2	(LPGM) HAS NO SUBCOMMAND OR ARRAY (CLUE).
3	(LPGM) HAS A PROBLEM WITH (CLUE).
4	(LPGM) FINDS AN EOF ON FILE (IN1).
5	(LPGM) FINDS (CLUE) SET TO (IN1) BUT ONLY (IN2) PERMITTED.
6	(LPGM) HAS NOT BEEN GIVEN ENOUGH DATA.
7	(LPGM) BUFFER SPACE EXHAUSTED.

No other values of IERR are supported.

2.4 Primitive Routines

The language processor also has eight primitive routines, GETCRD, SCAN, DELIM, AHEAD, FUNCTION-INDEX8, RVALUE, RDRV, and STRIN. Each of these primitive routines is discussed briefly below.

2.4.1 GETCRD

GETCRD reads the next record from the input file (usually unit 5), places each character as a separate member of array IC, and sets the index I1 to 1. GETCRD is called automatically by SCAN and many of the other higher-level language processor routines, so the user should not in general call it unless he wishes to skip whatever remains on the current record. When DO groups are being processed, GETCRD stores records in array BUF in /PGMFBC/ until the end of the outermost DO group has been found, and rereads records from this buffer throughout execution of the group.

2.4.2 SCAN

SCAN returns the first eight characters from the next contiguous character string in the variable IWRD. SCAN calls AHEAD and generates an error condition if the first character is not a letter, then calls DELIM. If the character string contains fewer than eight letters, the remaining spaces in IWRD are converted to blanks. If the character string contains more than eignt letters, the remaining ones are truncated. Any characters may follow the first one, except for the delimiters defined in SPAHR, which are listed in Chapter 2.4.3. Any of the delimiters will terminate the string.

2.4.3 DELIM

DELIM scans the remainder of the line for a delimiter, returns the delimiter's index value in IDLM, and returns an optional secondary index in IER. DELIM sets I1 to point to the location of this delimiter in the IC array, except in the case in which IER is set, in which case I1 is set to point to the character preceding. If one or more blanks precede one of the other defined delimiters, they are ignored. The delimiters and their IDLM and IER values are as follows:

Symbol	IDLM	IER
blank	0 or 1	0
,	2	0
=	3	0
.	0	-4
+	0	-5
-	0	-6
*	7	0
(	8	0
)	9	0
;	10	0
/	11	0
$	12	0

2.4.4 AHEAD (ITYP)

AHEAD returns the type of the next nonblank character. ITYP is set to zero if the next symbol is a digit (0-9), +1 if the next symbol is a letter (A-Z or underscore), and -1 if the next symbol is anything else.

2.4.5 FUNCTION-INDEX8 (LV,N,V)

FUNCTION-INDEX8 searches the array of eight-character names LV for one identical to V, and returns the index number. If V is not found in LV, a -1 is returned. If V is made up of blanks, a zero is returned.

2.4.6 RVALUE (X)

RVALUE reads the next item on the input record. If it is a number, its floating-point value is returned in X. If the next item is a character string, SCAN is called, and the character system variable table is searched for the corresponding number, which is returned. If no system variable has the indicated name, an error condition is raised.

2.4.7 RDRV (RV,N)

The RDRV routine reads in a series of numbers starting with the next item in the current line and continues for N numbers or until a delimiter $,), or

; is encountered, whichever comes first. RV is dimensioned as a single-precision floating point array.

On entry, a loop calls RVALUE to get the next number. If the delimiter following this number is an asterisk (*), then the number is used as a repetition factor and is assigned to the variable NIN. RVALUE is then called to read in the number following the asterisk, and this number is placed into the next NIN consecutive members of RV. If the delimiter is not an asterisk, then the number is placed in the next consecutive member of RV. If the last delimiter read was terminal, or if the RV array has been filled, RDRV returns. Otherwise, it begins the loop again.

2.4.8 STRIN (LEN,N,S)

STRIN reads in string S of maximum length LEN, and returns the actual length of the string read in N. The string may stand alone, in which case it begins with the next nonblank character found and is terminated by a $ or ;. The string may also be delimited by single or double quotation marks, in which case leading blanks and delimiting characters may be included, with the restriction that the semicolon must be avoided if the string is being entered within the range of a SPAHR IF or DO command.

3.0 CREATING A SPAHR COMMAND MODULE

Given the subroutines described above, the user can readily construct a SPAHR command module that can be called through the SPAHR system. Though SPAHR is sufficiently flexible to permit a variety of conventions, an optimal method is outlined below.

1) Each command module shall consist of a relatively small command processor subroutine that is called without arguments from the main command processor (subroutine DEMPAK). The only function of this subroutine is to initialize an array of control parameters and (optionally) data arrays and then to call the working routine, which will perform the actual task using the control parameters and data arrays.

2) Subroutine DFILE and the default values and constants from common block /CONSTC/ should be used whenever appropriate for all demographic calculations, file unit initialization, labels on tables, etc.

3) Storage to be shared between the command processor or main working routine and lower-level subroutines that is not passed in the form of formal parameters and is not already contained in one of the system common blocks should be passed in blank common. Storage to be used temporarily within a routine and not passed to other routines or to be saved between calls to or from that routine should be placed in /PROJC/.

4) When output is printed, all page skips should be generated by calls to subroutine TITL, so that the bookkeeping routines in the language processor can record output page numbers. It may also be useful to keep a running count of the number of lines printed in array LINES of block /TITLEC/, in the event that decisions about page skips must be made. Subroutine DFILE is available to initialize I/O files and to use flags.

5) SPAHR is designed to run either in batch or interactive timesharing environments. Therefore, some thought must be given to the appropriate response to an error condition. In batch mode, an error will cause the program to terminate. In an interactive job, however, it may be possible to make corrections and continue. Error conditions should therefore be treated by printing an informative message, followed by a call to subroutine DEMERV, followed by a return either to subroutine DEMPAK (to start the whole command over again) or to some other interactive recovery point. DEMERV is designed to stop all execution if called in batch mode and to return to the calling routine if called in interactive mode.

The remainder of Chapter 3 demonstrates the use of these conventions to construct a simple command module. This module, which will be named XAMPL, will simply read in some data and print it out again, along with the population in the current data block.

3.1 The Command Processor

Given the language processor interface routines described earlier, it is not difficult to construct a simple SPAHR command processor. The following items must be included:

1) An array of parameter keywords. In an IBM system, this should be a double-precision (REAL*8) array, initialized with the names of the single-valued parameters that may be manipulated upon execution. In this sample routine (XAMPL), the keyword array has the name KEYPAR and defines five options: a print file (parameter PRINT), a machine-readable output file (PUNCH), a detailed print output file (DETAIL), a terminal level print option (TERM), and an arbitrary parameter named VALUE. The array KEYPAR could be coded as

```
REAL*8KEYPAR(5)
DATA KEYPAR/5HPRINT,5HPUNCH,6HDETAIL,4HTERM,5HVALUE/
```

2) An array of parameter values. Both an integer and a single-precision floating point array should be declared for this purpose; they may be made equivalent. Either integers or floating-point values may be returned by GETPAR, depending on the first letter of the KEYPAR entry. The parameter value array could be coded as

```
DIMENSION RPPR(5),IPPR(5)
EQUIVALENCE (IPPR(1),RPPR(1))
```

3) A section defining default values for the parameters defined by the KEYPAR/IPPR/RPPR arrays. In the case of input/output files, default values are provided by the SPAHR block /CONSTC/. Assuming that our sample routine contains a copy of this block, we could code the default section as

```
C DEFAULT PRINT FILE
C DEFAULT CARD-IMAGE OUTPUT FILE
     RPPR(1) = ROPTS(4)
C DEFAULT DETAILED OUTPUT FILE
     RPPR(2) = ROPTS(5)
C DEFAULT TERMINAL (OR LOG FILE) PRINT CODE
     RPPR(3) = ROPTS(6)
C DEFAULT FOR PARAMETER 'VALUE'
     IPPR(4) = -1
     RPPR(5) = 0.0
```

4) A call to GETPAR to initalize the single-valued parameters.

```
CALL GETPAR(KEYPAR,5,RPPR,IPPR)
```

5) A series of calls to DFILE to initialize the input/output files that will actually be used, along with their use switches. The use switches are stored in an array named IOPT, the discussion of which is found under point 8) of this section. The ROPTS values are discussed in the section about /CONSTC/ (Chapter 5.4).

```
C SET THE REGULAR PRINTED OUTPUT FILE
     CALL DEFILE (RPPR(1),IFILE,IOPT(3)ROPTS(4),6)
C SET THE DETAILED PRINT FILE
     CALL DEFILE (RPPR(3),IDFILE,IOPT(2),ROPTS(6),IFILE)
C DEFAULT FOR THE MACHINE-READABLE OUTPUT
     CALL DEFILE (RPPR(2),IPUNCH,IPFLG,ROPTS(5),7)
```

This will code for a very simple command processor. For a more elaborate one, however, the user might read in some arrays specific to age and sex. In that case, space must be reserved for these arrays, assigning to them exactly as much space in the first two dimensions (age and sex) as is done in SPAHR common blocks and routines that work on age- and sex-specific arrays. Therefore additional FORTRAN statements must be written to facilitate the items in 6), 7), and 8) below.

6) An array of keywords and reserved space for the arrays to be entered.

```
REAL*8 KEYWRD(2)
DATA KEYWRD/6HARRAY1,6HARRAY2/
DIMENSION A1(20,2),A2(20,2)
```

7) A coded section to read the arrays if and only if they are present, and to assign default values otherwise. If the delimiter returned after the call to GETPAR was a 10, then by convention no more data follows. If another delimiter was returned, the next symbol must be one of the array names in KEYWRD. SCAN must be used to find this name. SCAN, in turn, places this name in the variable IWRD in /DATAC/, so /DATAC/ must be included in the nonexecutable section of code. Following the call to SCAN, INDEX8 is called to determine which of the array names (if any) was found, and finally RDLA is called to read in the arrays themselves. If the keyword was misspelled on input, INDEX8 returns a -1, and a call goes to DEMERV, which will terminate SPAHR in batch mode and return in interactive mode. To cover the interactive case, a RETURN statement follows immediately to return to the executive processor.

```
C HAVE WE REACHED THE END OF THE COMMAND?

     20 IF(IDLM.EQ.10)GOTO 100

C READ IN THE NEXT CHARACTER STRING

     CALL SCAN

C WAS IT ONE OF THE DEFINED ARRAY NAMES?
```

```
      K=INDEX8(KEYWRD,2,IWRD)
C IF IT WAS, GO TO THE APPROPRIATE SECTION
      IF(K.GT.0)GOTO (30,40),K
C IF IT WAS NOT, WRITE AN ERROR MESSAGE
      CALL DEMERV(2,8H,XAMPLE,IWRD,0,0)
      RETURN
C READ IN ARRAY1
      CALL RDLA(A1,LSEX,20,2,20,IFOUND)
      GOTO 20
C READ IN ARRAY2
40 CALL RDLA(A2,LSEX,20,2,20,IFOUND)
      GOTO 20
```

8) Finally, working routines to perform the actual computations desired. The TERM control value must be assigned to the first member of the IOPT array to control the terminal file output. ITERM is the terminal input/output unit number from /CCARDC/.

```
100 IOPT(1) = IPPR(4)
      CALL XAMPL1(IOPT,IFILE,ITERM,A1,A2,RPPR(5))
```

3.2 Working Routines

A SPAHR working routine should be coded as a subroutine that can be called with equal ease from the command processor and from a dummy program for development and testing. The dummy program must include a section to initialize all necessary data blocks. The sample routine developed here, although it calculates nothing of demographic interest, illustrates most of the features that should be coded into a SPAHR routine:

1) Page skips and automatic titles. The interface routine TITL is called to print a page skip and heading, but only if regular printing is to be done.

```
IF(IOPT(3).NE.0) CALL TITL(IFILE,8,8H<XAMPL1>,72,NAME)
```

2) Age and sex constants, provided by SPAHR. This section of the sample routine demonstrates the use of the constants in the /CONSTC/ blocks NA (the number of age groups) and NS (the number of sex groups).

```
      DO 100 J = 1, NS
      DO 100 I = 1, NA
      A2(I,J) = ALOG(DR(I,J))
100 CONTINUE
```

3) Age and sex labels, provided by SPAHR. This part of the program prints out the results, but only if appropriate. The labels on the printed output for age and sex come from 8-byte character arrays LAGE and LSEX in /CONSTC/.

```
      IF(IOPT(3).EQ.0) GO TO 900
      WRITE(IFILE,210) ((LSEX(J),J=1,2),K=1,3)
220 FORMAT(5H AGE ,10X,2HA2,8X,2HDR,5X,10HPOPULATION/
    1 1X,5HGROUP,3(4X,A6,6X,A4))
        DO 230 I = 1, NA
230 WRITE(IFILE,240) LAGE(I),(A2(I,J),J=1,2),
    1 (DR(I,J),J=1,2),(PP(I,J),J=1,2)
240 FORMAT(1X,A5,2F10.4,2F10.6,2F10.0)
```

4) Special terminal messages. In this segment the user writes an informative message to the terminal or log file, but only if appropriate. The first member of the IOPT array directs this activity.

```
C    WRITES OUT THE TERMINAL MESSAGE
      IF(IOPT(1).NE.0) WRITE(ITERM,910)
910 FORMAT(1X,28HXAMPL1 EXECUTED SUCCESSFULLY)
```

3.3 Adding the Command Module to the Executive Processor

All command modules must be able to be called through the SPAHR system. To do this, the executive processor, which is subroutine DEMPAK in the source code, must be modified at three points:

1) The array program name declaration (KPGM) and the counter, number of programs index (NPGM), must both be updated by 1, and an appropriate name must be installed in the DATA command:

```
C COMMAND STATEMENT DICTIONARY LIST
      REAL*8 KPGM(24)
      DATA NPGM/24/,KPGM/3HSET,5HPRINT,2HIF,4HELSE,2HDO,3HEND,
      * 7HOPTIONS,5HTITLE,4HDATA,7HPROJECT,7HLIFETAB,
1 6HMULDEC,8HANALYSIS,4HROOT,5HZEROS,5HLOTKA,6HSTABLE,
2 6HFACTOR,6HPOLATE,3HUSE,4HSAVE,3HOUT,4HHELP,5HINPUT/
```

must be changed to

```
C COMMAND STATEMENT DICTIONARY LIST
      REAL*8 KPGM(25)
      DATA NPGM/25/,KPGM/3HSET,5HPRINT,2HIF,4HELSE,2HDO,3HEND,
* 7HOPTIONS,5HTITLE,4HDATA,7HPROJECT,7HLIFETAB,
1 6HMULDEC,8HANALYSIS,4HROOT,5HZEROS,5HLOTKA,6HSTABLE,
2 6HFACTOR,6HPOLATE,3HUSE,4HSAVE,3HOUT,4HHELP,5HINPUT,
3 4HXAMPL/
```

2) The computed GOTO statement that controls branching to the various modules must be modified to go to a segment that calls our XAMPL module.

```
          GOTO (42,44,50,60,70,80,90,100,200,300,400,410,500,510,
         1     510,510,520,530,540,600,600,700,800,850), IPGM
```

must be changed to something like

```
          GOTO (42,44,50,60,70,80,90,100,200,300,400,410,500,510
         1     510,510,520,530,540,600,600,700,800,850,1000), IPGM
```

3) Then we install the call to the XAMPL command. The call to XAMPL must be followed by a "go to" statement label 15, which calls the SCAN routine to find the next command. The user should then install at some point a GOTO 15 statement code of the form

```
     C EXAMPLE MODULE
     1000 CALL XAMPL
          GOTO 15
     C DEFAULT FOR THE MACHINE-READABLE OUTPUT FILE.
          CALL DEFILE(RPPR(2),IPUNCH,IPFLG,ROPTS(5),7)
     C HAVE WE REACHED THE END OF THE COMMAND?
          20 IF(IDLM.EQ.10) GO TO 100
     C READ IN THE NEXT CHARACTER STRING
          CALL SCAN
     C WAS IT ONE OF THE DEFINED ARRAY NAMES?
          K = INDEX8(KEYWRD,2,IWRD)
     C IF IT WAS, GO TO THE APPROPRIATE SECTION
          IF(K.GT.0) GO TO (30,40),K
     C IF IT WAS NOT, WRITE AN ERROR MESSAGE
          CALL DEMERV(2,8H XAMPL,IWRD,0,0)
     C RETURN TO CALLING PROGRAM AT ONCE
          RETURN
     C READS IN ARRAY1
          30 CALL RDLA(A1,LSEX,20,2,20,IFOUND)
               GO TO 20
     C READS IN ARRAY2
          40 CALL RDLA(A2,LSEX,20,2,20,IFOUND)
               GO TO 20
     C
          100 IOPT(1) = IPPR(4)
               CALL XAMPL1(IOPT,IFILE,ITERM,A1,A2,RPPR(5))
               RETURN
              END
```

3.4 The Complete Example

The foregoing discussion has not shown all of the code that is needed to create the XAMPL routine, nor has it shown all segments of the code in their proper order. The complete working example is therefore reproduced here.

```
      SUBROUTINE XAMPL
C.XAMPL...................SPAHR............................
C
C SAMPLE COMMAND PROCESSOR FOR THE SPAHR SYSTEM.
C .........................................................
C
C <FREEIO> INTERFACE BLOCK
      REAL*8 IWRD
      INTEGER*2 IC,KSPK,KDIG,KLET
      COMMON /CCARDC/ IWRD,INUM,NUMDIG,IEXP,I1,I2,I1S,IDLM,IER,NSPK,
     1     ITSO,ITERM,INTERM,IRES,IC(133),KSPK(12),KDIG(10),KLET(27)
C
C CONSTANT DATA BLOCK
      REAL*8 LAGE,LSEX
      COMMON/CONSTC/IOPTS(10),NA,NAM1,NAP1,NS,LAGE(20),LSEX(3),
     1 AGE(20),WGE(20),IAGE(20),STPP(20,2),STPPT,STPPX,NMSTPP(10,2)
      DIMENSION ROPTS(10)
      EQUIVALENCE (IOPTS(1),ROPTS(1))
      REAL*8 KEYPAR(5)
      DATA KEYPAR/5HPRINT,5HPUNCH,6HDETAIL,4HTERM,5HVALUE/
      DIMENSION RPPR(5),IPPR(5)
      EQUIVALENCE (IPPR(1),RPPR(1))
C KEYWORDS FOR DATA ARRAYS
      REAL*8 KEYWRD(2)
      DATA KEYWRD/6HARRAY1,6HARRAY2/
C ARRAYS TO HOLD DATA
      DIMENSION A1(20,2),A2(20,2)
C DEFAULT PRINT FILE
      RPPR(1) = ROPTS(4)
C DEFAULT CARD-IMAGE OUTPUT FILE
      RPPR(2) = ROPTS(5)
C DEFAULT DETAILED OUTPUT FILE
      RPPR(3) = ROPTS(6)
C DEFAULT TERMINAL (OR LOG FILE) PRINT CODE
      IPPR(4) = -1
C DEFAULT FOR PARAMETER 'VALUE'
      RPPR(5) = 0.0
      CALL GETPAR(KEYPAR,5,RPPR,IPPR)
C SET THE REGULAR PRINTED OUTPUT FILE
      CALL DEFILE(RPPR(1),IFILE,IOPT(3),ROPTS(4),6)
C SET THE DETAILED PRINT FILE.  THE DEFAULT FOR THE FILE UNIT IS
C TAKEN TO BE WHATEVER WAS ASSIGNED TO THE REGULAR PRINT FILE, IF
C BOTH RPPR(2) AND ROPTS(6) WERE FOUND TO BE 0.0.
      CALL DEFILE(RPPR(3),IDFILE,IOPT(2),ROPTS(6),IFILE)
      SUBROUTINE XAMPL1(IOPT,IFILE,ITERM,A1,A2,VALUE)
C.XAMPL1...................SPAHR...........................
C
C SAMPLE WORKING ROUTINE FOR THE SPAHR SYSTEM
C..........................................................
C
C CONSTANT DATA BLOCK
      REAL*8 LAGE,LSEX
      COMMON/CONSTC/IOPTS(10),NA,NAM1,NAP1,NS,LAGE(20),LSEX(3),
     1 AGE(20),NGE(20),IAGE(20),STPP(20,2),STPPT,STPPX NMSTPP(10,2)
      DIMENSION ROPTS(10)
      EQUIVALENCE (IOPTS(1),ROPTS(1))
C
C CURRENT DATA BLOCK
      COMMON/DATAC/ITOC(28),IDATE,AOP,ASP,AKP,ACP,GFR,RNN,GRR,TFR,
     1      TP(3),TD(3),TB(3),ASDR(3,2),CDR(3),ASBR(3,2),CBR(3),
     2      SDR(3),SBR(3),GM(3),GOM(2,3),VLOG(3,3),NAME(18),LNM1,LNM2,
     3      FSC(20),FNM(20),FFX(20),BBT(20),BR(20),SEXR(20).PP(20,2),
     4      DD(20,2),BB(20,2),PPCT(20,2),VKX(20,2),SRX(20,2),DR(20,2),
     5      VL(20,2),VLL(20,2),VMX(20,2),QX(20.2),RX(20.2),AX(20,2)
     6      TX(20.2),EX(20,2),VDX(20,2),PI(100,2),DI(100,2).BI(100,2)
C
C FORMAL PARAMETER ARRAYS
      DIMENSION A1(20,2),A2(20.2)
C PRINTS OUT THE TITLE
      IF(IOPT(3).NE.0) CALL TITL(IFILE,8,8H<XAMPL1>,72,NAME)
C DOES SOME CALCULATIONS
      DO 100 J = 1, NS
      DO 100 I = 1, NA
      A2(I,J) = ALOG(DR(I,J))
  100 CONTINUE
C PRINT OUT THE RESULTS
      IF(IOPT(3).EQ.0) GO TO 900
C PRINT OUT THE COLUMN HEADINGS
      WRITE(IFILE,210) ((LSEX(J),J=1,2),K=1,3)
  220 FORMAT(5H AGE ,10X,2HA2,8X,2HDR,5X,10HPOPULATION/
     1 1X,5HGROUP,3(4X,A6,6X,A4))
      DO 230 I = 1, NA
  230 WRITE(IFILE,240) LAGE(I),(A2(I,J),J=1,2),
     1 (DR(I,J),J=1,2),(PP(I,J),J=1,2)
  240 FORMAT(1X,A5,2F10.4,2F10.6,2F10.0)
C END OF PRINT SECTION
C WRITES OUT THE TERMINAL MESSAGE
      IF(IOPT(1).NE.0) WRITE(ITERM,910)
  910 FORMAT(1X,28HXAMPL1 EXECUTED SUCCESSFULLY)
       RETURN
       END
```

4.0 SPAHR COMMAND MODULES: STRUCTURE AND FUNCTION

Chapter 4 describes the demographic analysis subroutines contained in SPAHR and their interrelationships. The various subroutines are grouped by command module, and within each module they are arranged in hierarchical order. The command processor is described first, then the subroutines it calls, and so on through successively lower levels. Unless otherwise indicated, all routines are SUBROUTINE subprograms.

The arguments IOPT, IFILE, ITERM, and NAME are used frequently in the subroutines described in this chapter. They refer to a three-member integer vector that controls, by convention, the production of printed output by the subroutine, the file unit number for printed output, the file unit number for log file or terminal output, and a 72-character string containing the name of the population being analyzed or some other identifying string in the printed output. In the routines here, NAME invariably is taken from /DATAC/.

In the descriptions in this chapter, formulae are often written in FORTRAN style. These loose copies of the formulae that are found in the routines differ in most cases only in the number of subscripts associated with arrays and the manner in which loops are defined. It is hoped that these formulae will allow the programmer to easily identify and understand particular sections of the actual code.

Descriptions of the various procedures used, expressed in ordinary mathematical notation, may be found in Volume I of this series.

4.1 Command: LIFETAB

The LIFETAB command calculates abridged life tables for both sexes. An abridged life table is one in which the age group intervals are more than one year. The method used is that described by Chiang (1968), with some simplifying assumptions. The results are stored in appropriate members of /DATAC/. Three subroutines are called when the command LIFETAB appears in the procedure file. These subroutines, LIFETB, LIFTB, and LIFTAB, are discussed in detail below.

4.1.1 LIFETB (MODE)

LIFETB is the command processor. It interprets the LIFETAB command statement by using GETPAR, and it calls working subroutine LIFTB. If offline graphs of life table functions are to be generated, LIFETB calls routine OFFPLT. (This option is not available in version 4.1.) MODE=1 implies a command level call; life tables will be calculated immediately following command interpretation. MODE=2 implies a subcommand interpretation call; the remainder of the command statement will be interpreted, but no call will be made to LIFETB before returning. MODE=3 implies a subcommand execution call; no attempt will be made to interpret a command line, but execution of LIFETB will proceed on the assumption that a MODE=2 call has already been made.

4.1.2 LIFTB (IPPR,RPPR,NPAR,MODE)

Subroutine LIFTB computes a simple abridged life table for each sex, calls DFILE to initialize input/output files for printing and card-image output files, and decides on the basis of the contents of the ITOC array whether there is enough data to compute a life table, and if so what kind. Depending on the data available, as indicated by the ITOC array, the life table will be computed from one of two data sources. If both the PP and DD arrays in /DATAC/ have been properly initialized, then PP and DD are used as primary input to LIFTAB. If PP and DD are not initialized, or if the RATES option was specified, the DR array is examined. If the DR array is properly set, then it is used as the primary input. If none of the appropriate data arrays have been properly set, then DEMERV is called to generate an error message and terminate execution. LIFTB also calls LIFTAB to calculate the life tables once for each sex, and then calculates a combined infant mortality index.

Schoen (1970) suggested use of the geometric mean of the age-specific death rates as a summary statistic for mortality independent of the population's age distribution, as an alternative to the more commonly used age-standardized death rate or expectation of life at birth. LIFTB calls routine ANALIF to calculate this geometric mean (commonly referred to as Schoen's del) if the results are to be printed out.

4.1.3 LIFTAB (IOPT,IFILE,JSX,NAME,PP,DD,DR,QX,VLLX,VLX,EX,TX,AX,RX)

Subroutine LIFTAB computes an abridged life table. If PP(1) < 0.5, it is assumed that PP and DD contain age-specific person-years of exposure and deaths respectively, and these are used to calculate the age-specific death rates DR. Otherwise, it is assumed that death rates were entered directly in DR. The age-specific death rates DR are converted to probabilities of death in the age interval QX by means of the formula

$$QX = WGE * DR / (1 + (1 - A)*DR)$$

where

WGE is the number of years in the age interval, and

A is the fraction of the age interval lived on average by those dying in it. Following Keyfitz & Flieger (1971), this is assumed to be 2.5 years for all five-year age groups after age 5 (A = 0.5) and 1.5 years for the 1-4 age group (A = 0.375), and A is assumed to be 0.07+1.7*DR for the age group less than 1.

From QX, the number VLX surviving out of 100,000 births to the beginning of age interval I is computed as

$$VLX(I+1) = (1 - QX(I)) * VLX(I)$$

and VLLX, the person-years lived in each age group by the cohort, is estimated as

VLLX(I) = VLX(I+1) * WGE + A * (VLX(I) - VLX(I+1))

The TX array represents the total person-years lived from age group I onwards and is accumulated backwards from VLLX as

TX(I) = TX(I+1) + VLLX(I)

except for the final age group NA, which uses the approximation

TX(NA) = VLX(NA) / DR(NA)

The expectation of life at the beginning of each age group is placed in the EX array as

EX(I) = TX(I) / VLX(I)

The input DR values, based on real data, are then replaced by the life table death rates as

DR(I) = (VLX(I) - VLX(I*1)) / VLLX(I)

RX is not used, but is coded in this version to allow for planned expansion of the capability of the program for using the Keyfitz and Flieger (1971) algorithm to refine the estimates.

4.2 Command: MULDEC

This command calculates abridged multiple decrement and associated single decrement life tables. A multiple decrement life table is one in which the various causes of death are analyzed within the regular life table, and their relative importance in the presence of other competing risks is shown. An associated single decrement life table is one in which the incremental importance of the cause of death of interest is analyzed by calculating the life table in its absence. The construction and use of such tables are discussed by Chiang (1968) and Preston, Keyfitz, and Schoen (1972). Three subroutines are called when MULDEC appears as a command in the procedure file: MULDEC, MULDC, and MULTIP.

4.2.1 MULDEC (MODE)

Command processor MULDEC uses GETPAR to initialize the control parameter arrays IPPR/RPPR and calls working routine MULDC. The interpretation of the MODE parameter is the same as in the LIFETB routine (Chapter 4.1.1).

4.2.2 MULDC (IPPR,RPPR,NPAR)

Subroutine MULDC initializes the input/output files through appropriate calls to DFILE. MULDC also examines the ITOC array to determine if sufficient information is available to perform a multiple decrement analysis by checking for the presence of the VLX array (which is generally set by a call to the LIFETAB command) and the DRC array. If either VLX or DRC is missing, DEMERV is called. Because the calculations to follow require that raw population and death data be present, these are copied from PP and DD into local arrays PPP and DDD. If PP and DD are not properly initialized, PPP and DDD are filled with dummy numbers based on the contents of DR in /DATAC/.

A loop that steps through each cause of death is then entered. Deaths from each cause are copied from DRC in /KOSC/ or calculated from DRC and PPP depending on whether DRC contains raw numbers or rates. MULTIP is then called to calculate the associated single decrement (ASD) life table and multiple decrement life table columns for each cause. On exit from MULTIP, the appropriate section of DRC has been replaced with death rates.

The array RV is then initialized with a set of summary measures for each cause and sex.

RV(1,*,*)	is set to the sex-specific crude death rate for each cause (total deaths divided by total population).
RV(2,*,*)	is set to the sex-specific, age-standardized death rate for the first standard population in STPP.
RV(3,*,*)	is returned as the ASDR for the second standard population.
RV(4,*,*)	contains the years of life lost to each cause, calculated as the difference between the expectations of life at birth in the original and ASD life tables.
RV(5,*,*)	is set to the years of working life lost to the cause, calculated as the difference between the VLLX arrays for the ages 15 to 65, divided by 100,000.
RV(6,*,*)	is the difference between the geometric mean death rates for the total age-specific death rates in the original and ASD life tables. This variation in Schoen's <u>del</u> (see 4.1.2) avoids the pitfalls inherent in the relatively large number of causes for which part of the age range is likely to show zero probability of death.

If offline graphs of the cause-specific death rates are to be generated, OFFPLT is called to generate the necessary file.

4.2.3 MULTIP (IOPT,IFILE,JSX,NAME,KLOS,PP,DD,DDC,DR,DRC,DRNC,VL,VLC,VLA,VLLA, EXA)

MULTIP is the central working routine for the MULDEC command and performs most of the calculations. First, the deaths from the cause of interest DDC are subtracted from the total deaths DD, and the result (local array DDNC) is used in a call to LIFTAB to calculate the life table. The survivor, person-years lived, and expectation of life arrays from this calculation are returned in VLA, VLLA, and EXA, respectively.

The survivors column in the original life table VL is used in conjunction with the mortality information to calculate the numbers of survivors of each age who will eventually die of the cause of interest as

$$VLC(I) = \sum_{J=I}^{NA} (VL(J)-VL(J+1))*DDC(J)/DD(J)$$

where

VLC is the number of people arriving in age group I who will eventually die of the cause of interest.

VL is the total number of people surviving to the beginning of age group J.

DDC is the number of people in the original data dying of the cause of interest in age group J.

DD is the total number of deaths in age group J in the original data.

The death rate for the cause of interest is then calculated as

$$DRC = DDC \ / \ PP$$

JSX is the sex index, used primarily to enable the printed output to be properly labeled.

4.3 Command: PROJECT

The command PROJECT generates projections of the population, total deaths, deaths by cause, births, and assorted measurements related to the structure of the population as it evolves over time. It also possesses a number of sophisticated features for projecting changing vital rates over time as a function of exposure to various risk factors, as well as a facility to estimate gradual changes in birth and death rates between two points in time by linear interpolation. The routines in the PROJECT command group can operate in two projection modes (PERIOD and COHORT) and two special operating modes (RATES and RANDOM).

PERIOD mode projections generate the projected cross-sectional structure of the population at specific intervals. Most population projections performed by census bureaus and other demographic prognosticators are of this type. This projection mode is handled by subroutine PRJPER.

COHORT mode projections generate an estimate of the size and vital experiences of a single group of people as they progress forward through time and age. A cohort projection has more in common with the construction of a life table than with a period projection. Cohort projections are performed by the subroutine PRJCOH.

RATES mode has the ability, through the use of interpolations between preset values or calls to the ADJUST routines, to vary the age-specific mortality rates over time. In this mode, projection of the population and deaths is omitted. Instead, the changing death rates are analyzed in detail by calls to the LIFETAB and MULDEC command routines. This output mode, unlike the others, will cause the data in the /DATAC/ and /KOSC/ blocks to be altered. It may not be selected if RANDOM mode is also selected. It is implemented in both cohort and period mode through special branches in both PRJPER and PRJCOH.

RANDOM mode results in a Monte Carlo projection. An initial normal projection is generated. This is followed by a number of projections of the same population and rates, with the results perturbed by the use of a random number scheme, rounded to integer values. The loop controlling the number of projections uses the control variable IREPT in /PROJC/. A number of the subsidiary routines in the PROJECT group take special action (such as calls to the random number generator) whenever IREPT is greater than 1, thus signifying that RANDOM mode is in effect. RANDOM should not be selected for COHORT or RATES mode projections.

The command PROJECT, when invoked in the SPAHR procedure file, calls several other subroutines in addition to those already mentioned. These include PRJECT, PRJPR1, PRJMC1, PRJAC1, PRJSRV, PROJ, PRJKOS, DEPAR, RANBIN and INDOS. Only selected subroutines are used in any given invocation of the command PROJECT.

4.3.1 PRJECT

PRJECT is the command processor for the PROJECT command. It is a relatively large routine that interprets the remainder of the PROJECT command statement by using GETPAR. If the return from GETPAR is signalled by a delimiter other than a semicolon, a loop that calls SCAN and INDEX8 to detect array and subcommand names is entered. The loop then directs the arrays and subcommands to special processing sections as it finds them. Arrays are read in using RDLA or RDRV, while the subcommands are processed through calls to the subcommand processors ADJUST, LIFETB, and MULDEC.

When the end of the statement is found (denoted by the semicolon return value of 10 in IDLM), PRJECT sets the initial population array (PSTRT), death rates (DRZ), birth rates (BRZ), and the interpolation indices. If any ADJUST calls indicate that the BACKGROU option was selected, ADJDUR is called to calculate the effects on mortality of the long-term background level exposures and to subtract these effects from the initial death rates DRZ, final death rates DRL, and appropriate cause-specific death rates in DRC if these were included in the original input.

PRJPER is then called if a period projection is to be performed, and PRJCOH if cohort projections have been selected. Following a return from PRJPER (if PRJCOH is not scheduled for a subsequent call), PRJECT may optionally save the population and vital rates as they have been projected.

4.3.2 PRJPER

Routine PRJPER performs a period (or cross-sectional) population projection. The array that stores the projected population (POP) is initialized from PSTRT, and the birth and death rates to be used for the projection (BRX and DRX) are initialized from BRZ and DRZ, respectively.

A loop that steps through time from the beginning to the end of the projection in five-year intervals is entered at this point. During each step, the interpolation control parameters are examined, and the baseline birth and death rates are altered as indicated. This alteration is done in the following way: If ISTDR (the initial interpolation year) is found to be less than the terminal year of the current projection interval, the midpoint of the current interval is used in conjunction with ISTDR and ISTPDR (the final interpolation year for death rates) to allow a linear interpolation between the initial and final death rates DRZ and DRL, using the formula

$$\mathrm{DRX} = (\mathrm{DRL} * \mathrm{X}) + \left[\mathrm{DRZ} * (1-\mathrm{X})\right]$$

where

X   is (YR-ISTPDR) / (ISTDR-ISTPDR), and

YR   is the midpoint of the projection step.

If the beginning of the projection interval is found to exceed ISTPDR, DRX is set to DRL, and the process is terminated for the rest of the projection by setting ISTDR to a number greater than the final year of the projection. A similar procedure is followed for interpolating the birth rates.

If the ADJUST programs were invoked in PRJECT (as indicated if NPUSH were found > 0), routine ADJDUR is called to adjust mortality rates further on the basis of the exposures defined in the ADJUST subcommand.

If any of the vital rates have been altered, or if the loop is making its first pass, routine PRJSRV is called to calculate the projection ratios needed to project the population forward through the time interval. The routine PROJ is called to take the population in POP(*,*,K) and project it forward into the next interval POP(*,*,K+1), where K is the index associated with the current interval. PROJ also returns the age-specific deaths during the interval in DTH, and the total births in BTH.

PYRMID can be called to print out the current age pyramid, and if multiple causes of death are part of the projection, PRJKOS is called to partition the projected deaths in the interval DTH by cause. The deaths by cause and sex are returned in DTC. Finally, a set of summary statistics is calculated and stored in array VSTAT. Each VSTAT row is defined below. Each item is generated for each sex and projection interval.

Row	
3	Crude Death Rate (CDR)
4	Crude Birth Rate (CBR)
5	Crude Rate of Population Change (CBR-CDR)
6	Proportion of population < 15 years of age
7	Proportion ≥ 65 years of age
8	Dependency ratio $\frac{(\% < \text{age } 15) + (\% \geq \text{age } 65)}{\% \text{ ages } 15\text{-}64}$
9	Expectation of life at birth
10	Expectation of life at age 10
11	Expectation of life at age 65
12	Age-standardized death rate (using the first standard population in the STPP array)

In addition, the array FRI is assembled for the following summary arrays that are not sex-specific:

Row	
1	Net Reproduction Rate (NRR)
2	Gross Reproduction Rate (GRR)
3	Total Fertility Rate (TFR)

At intervals specified by the user, but not greater than 50 years, PRJPR1 is called with MODE=1 to print out the summary of results for each projection interval. The 50 year limitation is required because only 11 columns are available in the POP array, and the width of a line printer page will not allow more. If the projection continues beyond the print interval, the Kth column of the POP array is copied to the first column, K is reset to 1, and the loop continues.

At the end of the projection, PRJMBT is called to generate an analysis of the aggregate morbidity levels during the projection, and a block of code calculates the incidence of selected diseases based on the numbers of deaths from these diseases. PRJPR1 is then called with MODE=2 to print out the results of these aggregate summary calculations.

If the RANDOM parameter is not mentioned, RPPR(30) enters as zero, so NREPT is set to 1 and only a single projection is performed. If RPPR enters as -1, which will happen if RANDOM is mentioned as a switch, NREPT is set to 2, as is the variable MREPT, and a single projection is performed, but with random perturbations. If RPPR(30) comes in with any magnitude greater than 1, NREPT is set to RPPR(30)+1, MREPT is set to 1, and the indicated number of projections will be performed. The first projection, however, will be a normal deterministic projection. Only the second and successive projections will be random.

When PRJPR1 is called, a record is written out on unit 1 if either the ACCUM parameter (IPPR(9)) or the RANDOM parameter (RPPR(30) = 0 and IREPT > 1) is in effect. These records consist of the data that are printed out in PRJPR1 and are used by PRJMC1 and PRJAC1 to prepare their respective summary tables.

4.3.3 PRJPR1 (MODE,K,N1,IDENT)

Routine PRJPR1 prints out the summary results of the period projections generated by PRJPER. The parameter MODE determines whether the interval-specific output (MODE=1) or the aggregate summary output (MODE=2) will be generated. IDENT is a string of length N1 that is used in the call to TITL to generate page titles.

4.3.4 PRJMC1 (N,K)

PRJMC1 is the postprocessing routine for the Monte Carlo simulation (option RANDOM) execution of the PRJECT/PRJPER routine. If the RANDOM option was selected in PRJPER, then file 1 contains a series of blocks containing the contents of the summary arrays POP, DTH, BTH, etc., up through interval index K. PRJMC1 rewinds file 1 and calculates the mean, standard deviation, and variance-mean ratio for every item in the table. Then PRJPR1 is called to print out the results.

4.3.5 PRJAC1

Routine PRJAC1 is the postprocessor for the ACCUM option. When called, it rewinds file 1 and reads back the records for each projection, adding them into the storage arrays (POP, DTH, etc.) from which the data were originally transferred onto the file. Rates and life table items are recalculated based on this pooled data, and PRJPR1 is called to print out the pooled projections in the format of an ordinary projection.

4.3.6 PRJSRV (IOPT,IFILE,NAME,DR,DRAT,BR,SEXR,SR,FF,RV)

Routine PRJSRV calculates the quantities needed to perform a population and mortality projection. For each sex, a life table is calculated based on the input death rates DR and a call to LIFTAB. On the assumption that populations over short intervals may be approximated by a life table age distribution, survival ratios for each age group I are calculated as

$$SR(I) = VLLX(I)/VLLX(I-1)$$

where

VLLX is the life table person-years column.

The expected proportion in age group I dying during the projection interval during the passage from age group I-1 to I is calculated as

$$DRAT(I) = (VLX(I) - VLX(I+1)) / (VLX(I-1) - VLX(I+1))$$

where

VLX is the life table survivors column.

The birth ratios FF, used to calculate total births in a projection cycle contributed by the females starting in age group I at the beginning of the cycle, are given by

$$FF(I) = 2.5*(BR(I) + (SR(I+1)*BR(I+1)).$$

The array RV is returned with a number of intermediate quantities. Many of its members are reserved for future expansion. As currently implemented,

RV(1,J)	contains the expectation of life at birth for sex J;
RV(2,J)	contains the expectation of life at age 10 for sex J;
RV(3,J)	contains the expectation of life at age 65 for sex J;
RV(4,J)	contains the ratio of person-years lived in the first year of life to births in the life table for sex J;
RV(15,1)	contains the Net Reproduction Rate (NRR);
RV(15,2)	contains the Gross Reproduction Rate (GRR);
RV(16,1)	contains the Total Fertility Rate (TFR).

4.3.7 PROJ (IOPT,IFILE,NAME,SR,BR,PP1,PP2,DD,BB,DEATHR,SEXR)

Routine PROJ takes the population at the beginning of the projection interval PP1 and projects in forward into PP2. For all age groups except the first and last,

```
PP2(I) = SR(I) * PP1(I-1)
```

where

SR is the survival ratio calculated in PRJSRV.

If the RANDOM option is in effect, the population is then perturbed by using the normal approximation

```
PP2(I) = PP2(I) + RANORM(0)*SQRT(PP2(I)*(1-SR(I))
```

The first age group (0-5) is calculated in several stages. First, total births for each age group I are calculated as

```
BBS(I) = 2.5 * BR(I) * PP1(I) + PP2(I)
```

These are then summed, and the total is multiplied by SR(1) to yield PP2(1). If the RANDOM option is in effect, the births are perturbed as

```
BBS(I) = BBS(I) + RANORM(0)*SQRT(BBS(I))
```

The deaths by age group among those dying in the interval are then calculated using the DEATHR array (which was called DRAT in the discussion of PRJSRV) as

```
DD(I) = (PP1(I)-PP2(I+1))*(1-DEATHR(I+1)) + (PP1(I-1) - PP2(I))*DEATHR(I)
```

If the RANDOM option is in effect, the allocation between age groups is made using routine DEPAR.

4.3.8 PRJCOH

PRJCOH is the working routine that oversees cohort projections. It should be called from PRJECT only if the COHORT option has been selected. A cohort projection differs from a period projection in that the structure of a population is not projected over time, but a selected initial age group is followed through age and time. In a period projection, the predicted deaths by age group and time interval are calculated by combining members of two cohorts. PRJCOH calculates the numbers of deaths in each cohort for each age interval, but mixes projection intervals in order to do so.

After initializing the vital rates and control variables, PRJCOH enters a loop that steps through each successive age group of the initial population and each birth cohort generated by the preceding call to PRJPER. Then a second loop that steps through time in five-year intervals from the birth interval of the cohort to its final (open ended) interval is entered. This second loop sets the DRX and BRX arrays to their appropriate age- and time-specific values by using the same interpolation formulae and calls to ADJDUR as would be done in a period projection. The loop then calls PRJSRV to generate cohort-specific projection ratios. The cohort is then projected through to extinction using the methodology described in PROJ. If the cohort had already been born at the time the projection started, (i.e., if it was a member of the initial population) it is back-projected to its interval of birth as well. PRJKOS is called to estimate distribution of deaths by cause.

4.3.9 PRJKOS (IOPT,IFILE,NAME,DD,DR,DCT,NKOS,LKOS,DRC,IMD)

Routine PRJKOS allocates an age-specific array of deaths by cause, given a set of total death rates and death rates by cause. Total deaths DD are copied into the temporary array DDD, and total death rates from DR are copied into DRR. Then a loop that steps through the first NKOS-1 causes of death is entered. The index array IDM determines whether a particular cause of death is a primary group or a sum of primary groups, hereafter called a supergroup. IMD(K) is set to 0 if cause K is a supergroup, to the K value of its supergroup if it is a primary group belonging to a supergroup, and to -1 if it is a primary group that is not a member of a supergroup.

If cause K is a primary group, the total age-specific death rates DR and the cause- and age-specific death rates DRC are used to compute the deaths DBC for age I, sex J, and cause K as

DBC(I,J,K) = DDD(I,J) * DRC(I,J,K) / DRR(I,J)

If the RANDOM option is in effect, the deaths are randomly allocated by cause by a sequence of binary random partitions using subroutine DEPAR, with DDD storing on return the remaining unallocated deaths, and DRR storing the corresponding fraction of the death rate DR.

If IMD(K) > 0, then the deaths allocated to cause K are added into the indicated supergroup after they have been computed. The NKOSth group is treated as a supergroup in which all primary groups following the one with the name _BASELN_ in the group name array LKOS are accumulated.

4.3.10 DEPAR (DT,DC,RT,RC,ISEED)

Routine DEPAR performs randomized partitions of deaths from the cause of interest DC in total deaths DT. The expected proportion of deaths in the group of interest is first calculated as

$$P = RC / RT$$

where

RC is the death rate from the cause of interest, and

RT is the total death rate.

Subroutine RANBIN is then called to allocate the randomized deaths for the cause of interest. On return, DT is decremented by DC, and RT is decremented by RC.

4.3.11 RANBIN (T,S,PP,ISEED)

Routine RANBIN randomly allocates S successes in T trials with underlying probability PP. The algorithm used was developed by Dr. Michael Ginevan, who also coded most of the routine.

The expected number of deaths in the group of interest is first calculated as

$$E = T * P$$

where P is the expected proportion of deaths in the groups of interest. If E is large, (i.e., >50) a normal approximation method is used to apply a random perturbation to the expected number as

$$S = E + SQRT(E*(1-PP)*RANORM(0))$$

where RANORM(ISEED) is a random unit normal deviate.

If $t < 50$, a uniform random number is generated using subroutine RANDU, and a loop is entered that calculates the probability of successive numbers of deaths from the cause of interest (or all other deaths, whichever is less) using the binomial recursion formula and accumulating probability until it exceeds the random number. The number of successes at this point is returned as S.

The argument ISEED is passed through as the argument to the uniform random number generator RANDU, which is called by RANORM. Unless the current call is the first call to any of the random number routines, ISEED will have the value 0. In the first call ISEED contains a copy of the input parameter SEED, which is used to initialize the random number sequence. RANDU resets ISEED to 0 before returning.

4.3.12 INTDOS (RFILE,NAME,IYR,LKOS,DCSUM,DOSINT,P1,P2)

Routine INTDOS accumulates integrated person-units of exposure to the hazards defined in calls to the ADJUST subcommand processor. P1 and P2 are the total population sizes at the beginning and end of the time interval. DOSINT is an array that maintains the cumulative integrated dose. It contains an entry corresponding to every cause of death recognized in the projection, to a maximum of NKM. NKM is the maximum possible number of causes of death, defined in block /KOSC/.

When called, INTDOS enters a loop that steps through successive levels of the pushdown stack in block /PUSHDC/. /PUSHDC/ stores the control parameter section of /ADJSTC/ maintained for successive ADJUST invocations. Each level is associated with an indicator that tells whether the dose in question derives from a constant exposure level or from a column in the DOSE array in /ADJSTC/.

If the exposure is a constant level, its value is stored in the stack entry corresponding to variable PEX in /ADJSTC/. This value is multiplied by 2.5*(P1+P2) and added to each member of DOSINT that represents a cause of death referenced by the current level of the pushdown stack.

If the dose comes from the variable array DOSE in /ADJSTC/, then the population in each year of the projection interval is estimated using the straight-line method, and the integrated dose is accumulated for each year individually. If printed output is called for, the accumulated values for the deaths in each cause (DCT), the integrated dose DOSINT, and their ratio (DCT/DOSINT) are printed out for each cause and sex.

4.4 Subcommand: ADJUST

This section describes a set of routines that process the ADJUST subcommand, which is called from the PROJECT command. This set of routines reads in a series of exposure estimates for environmental hazards (e.g., radiation, cigarette smoke, and a cigarette-derived air pollution model) and calculates the degree to which these hazards increase mortality rates.

The ADJUST routines are coded to operate in two projection modes (period and cohort) and two risk modes (absolute and relative).

PERIOD	mode assumes that the exposure history applies equally to all age groups, subject only to modification by age at onset and/or termination of exposure considerations. The parameter listed in the discussions below as IYR will always refer to the current year relative to the beginning of the exposure history. PERIOD mode is indicated when the LADJC parameter in /PROJC/ is set to .FALSE..
COHORT	mode assumes that the death rates being calculated refer to a single cohort whose birth interval relative to the beginning of the exposure history is passed in IYR. Each age group is advanced in time by 5 years each time the calculations regarding its duration of exposure are performed. COHORT mode is indicated when the LADJC parameter in /PROJC/ is set to .TRUE..
ABSOLUTE	mode implies that the increased risk of death calculated is not dependent in any way upon the existing death rates. ABSOLUTE mode is signaled when IR in /ADJSTC/ has the value 1.
RELATIVE	mode implies a dependence between the preexisting death rates and those calculated by the ADJUST routines. RELATIVE mode is indicated when IR in /ADJSTC/ has the value 2.

Six subroutines may be invoked when the subcommand ADJUST appears in the procedure file: ADJUST, ADJDUR, ADJAIR, ADJRAD, ADJR and NLMOD. Not all of these subroutines will be used in a single appearance of the subcommand ADJUST.

4.4.1 ADJUST

This is the command processor for the ADJUST subcommand. Its interpretation of the subcommand begins with a call to GETPAR to initialize the single-valued options. Based on these options, particularly on the MODEL parameter (IPR(3) in /ADJSTC/), it initializes the default coefficient arrays and cause names associated with the current model into the temporary arrays CCT, RLAT, RLPLA, XMRBT, and LLKOS from the default storage arrays BETA, LATENT, LPLAT, XXMRB, and LLKOS in /ADJRDC/, respectively. If a semicolon delimiter was not detected by GETPAR, a SCAN/INDEX8 loop is entered that uses RDLA and RDLA2 to modify the temporary arrays and the DOSE array with standard SPAHR array operations.

The temporary coefficient arrays, as modified, are then copied into the next unused sections of their corresponding permanent arrays: CC, LATNT, and

LPLT (all in /ADJSTC/), and XMRB and LKOS (in /KOSC/). The DOSE array is initialized from the DOSET array by adding to each member the sum of the appropriate constant dose term (if entered as a single-valued option), the time-specific whole body exposure (for radiation models only), and the various classified time-specific doses.

If, when the permanent coefficient arrays are set, the current model has defined more than one additional cause of death, a special cause group named _EXCESS_ is created, and the IMD indices are set so that all other causes defined in the current model point to _EXCESS_. This will have the effect of causing all deaths calculated for the other cause-of-death groups defined to be accumulated by various other parts of the SPAHR program into _EXCESS_. If only one new cause of death was defined in the current model, the IMD index is set to -1. (The IMD index vector is described in the section on /KOSC/, Chapter 5.6.) Just before returning, the pushdown stack pointer NPUSH is incremented, and the first 45 members of the control block /ADJSTC/ are stored in the stack. There is room for 5 separate calls to ADJUST in the pushdown stack.

4.4.2 ADJDUR (IOPT, IFILIN,NAME,DURAT,DRIN,DROUT,LCALL)

Routine ADJDUR determines exposure indices and returns the total increment in mortality rates due to those exposures. A loop is entered that steps through each level of the pushdown stack that has been initialized by a call to ADJUST. At each level, the duration since onset of exposure is calculated as

$$DUR = DURAT + DURP$$

where

DURAT is the number of years that have elapsed since the projection began in period mode or the years elapsed since the birth interval of the cohort in cohort mode, and

DURP is the difference in years between the initiation of exposure in the currently adjusted stack level and the start of the projection.

If the DUR value indicates that exposure for the current ADJUSTment has not yet begun, or if constant exposure has prevailed for over 97.5 years, that adjustment is skipped. In the latter case, the mortality increment is assumed constant after 97.5 years and allowed to remain at the last calculated value. The local variable XCOH is added to DUR for this test. XCOH is set to 0 in period mode and 90 years in cohort mode.

If the BACKGROU option was selected for the current NPUSH level, further processing is omitted after the initial ADJDUR call, because a changed value would never be returned. During that first ADJDUR call, the mortality increment calculated is subtracted from the initial mortality rates, thereby dividing initial mortality into two categories: that which is due to the background exposure to the hazard of interest, and that due to all other factors. The subtraction is repeated at each call if the LADJ parameter in /PROJC/ indicates that interpolation of the total mortality rate is in effect.

ADJDUR then sets the print indices (which may be selected independently during each ADJUST invocation) and calls either ADJAIR or ADJRAD, depending on the type of exposure being considered.

When the stack loop is completed, another loop is entered that sums the causes of death influenced by the preceding loop into DROUT. To DROUT is then added DRIN to give the final total mortality rate increment.

4.4.3 ADJAIR (IOPT,IFILE,NAME,DUR,DR,DRC)

Routine ADJAIR implements the cigarette-indexed model proposed by Lundy (Lundy & Grahn 1977). Only constant exposure levels modified by age and time of onset of exposure are handled. The increment to mortality DRC is computed in absolute mode as

```
DRC(I) = PEX * CRA(1) * EXP(CRA(2)*AGE(I)) / (1 + CRA(3) * EXP(CRA(4)*DURAT))
```

and in relative mode as

```
DRC(I) = PEX * CRR(1) * (DR(I)**CRR(2)) / (1 + CRR(3) * EXP(CRR(4) * DURAT))
```

where

CRA,CRR	are coefficients specific to the absolute and relative risk modes, respectively,
AGE(I)	is the midpoint (in years) of age interval I, passed in /CONSTC/,
PEX	is the constant exposure level in cigarette-per-day equivalents, and
DURAT	is the lesser of DUR (the input duration since onset of exposure) and AGE(I) - MINAGE(I) (the difference between current age and the earliest possible age at exposure). If DURAT is negative, the ADJAIR calculation is omitted. In cohort mode, DUR is the onset of exposure relative to the interval of birth of the cohort, so that DURAT is incremented by five years on each pass through the age loop.

4.4.4 ADJRAD (IOPT,IFILE,NAME,IYEAR,DR,LKOS,DRIN,IYR,KSR,KST)

Routine ADJRAD implements the absolute risk radiation model of the Reactor Safety Study (USNRC, 1975) and a relative risk model derived from the BEIR report (NAS, 1972), the latter with consideration to some information incorporated into the NRC model. In addition, a second set of radiation models from the 1980 BEIR report (NAS, 1980) is given. This set of radiation models includes a linear, a linear-quadratic, and a pure quadratic model.

On entry, ADJRAD sets the parameter KPLAT (based on the LPLAT parameter of the ADJUST subcommand) to 1 or 2, depending on the nature of the plateau period desired, and calls ADJR to perform the actual computations. In absolute risk mode, the quantities returned in DR are the incremental death rates, and ADJRAD returns at once. In relative risk mode, the DR array returns the factors by which the DRIN values must be multiplied in order to generate the increment in mortality rates, and ADJRAD generates the increment.

4.4.5 ADJR (IOPT,IFILE,NAME,IYEAR,DR,LDOSE,DOSE,IDOSE,IYR,IYRB,LTYPE,B,LATENT, LPLAT,LKOS,NK,MINAGE,MAXAGE)

Routine ADJR implements a general risk model algorithm based on the accumulation of age-specific risk for a period of time following exposure to a hazardous agent. ADJR is based on the method employed in deriving the risk estimates for the 1972 BEIR report (NAS, 1972), the results of which report it duplicates exactly with the exception of one number (out of nearly 100) that appears to be a typographical error in the BEIR tables. ADJR is also based on methods of the BEIR 1980 committee report (NAS, 1980).

In this model, it is assumed that for each annual increment of exposure, a proportional increment of risk occurs. This increase in exposure does not appear as an increase in the risk of death, however, until a certain length of time (the latency period) has passed. The risk then rises by an elevated (and constant) factor for a number of years (the plateau period) and finally returns to 0. When exposure occurs over several years, the increased risk in the current year is assumed to be made up of the increases due to the exposure in each prior year, which are treated independently.

ADJR implements the model in the following way. First, the arrays DR (to contain the age-, sex-, cause-specific output) and DRQ (to contain the same data subdivided in addition by age at exposure) are cleared. Then a loop that steps through each year of age is entered. For each age, IAG, the condensed age group index (IANOW), and the proportion of this condensed age group that is spanned by a single year (FAC) are calculated. Another loop is then entered that steps through each age at exposure and determines for each cause of death whether that year falls within the plateau period. If it does not, calculations stop. Calculations are all based on the midpoint of the current year and refer to the entire preceding year. Thus, the dose assigned (variable DOSEX) is computed for most years IY as

```
DOSEX = 0.5 * (DOSE(IY) + DOSE(IY-1))
```

Where the year of birth is concerned an additional complication arises. On average, a child less than one year of age is 6 months old. However, the child was also in utero for 9 months preceding birth. If the child was in fact born 11.9 months ago, much of its total exposure could have been accumulated in the year prior to the current one. Reference to a Lexis diagram (the use of Lexis diagrams is discussed by Keyfitz, 1968) shows that the expected exposure in utero can be calculated as

```
DOSEX = (0.125*DOSE(IY)) + (0.59375*DOSE(IY-1)) + (0.03125*DOSE(IY-2))
```

While the coefficients in the case of adult exposure (0.5) sum to 1 (for a whole year of exposure), the coefficients for exposure in utero sum to 0.75, reflecting the period of gestation.

The effect on the death rate for the current age is then calculated as

```
EFFECT = DOSEX * B(IAX,J,K) * FAC
```

where

B	is the effect coefficient,
J	is the sex index,
IAX	is the condensed age-at-irradiation index. =1 for in utero age group =2 for 0-9 years of age =3 for 10-19 years of age =4 for 20+ years of age

Separate values for DOSEX and EFFECT are calculated for each age group at exposure IAX and added into DR and DRQ. If a nonlinear model is employed, the subroutine NLMOD is called.

4.4.6 NLMOD (EFFECT,DOSEX,NONLIN,I,J,K,FAC,B,NK)

Subroutine NLMOD determines the excess mortality for the linear-quadratic and the pure quadratic models of the BEIR 1980 report (NAS 1980). NONLIN is set to 1 for the linear-quadratic and to 2 for the pure quadratic model. For the linear-quadratic model, the death rate is estimated as

```
EFFECT = ((B(IAX,J,K)*DOSEX)+(COEFD*(DOSEX*DOSEX)))*FAC
```

where

```
COEFD = 0.008614*B(IAX,J,K)
```

The death rate for the pure quadratic model is estimated as

```
EFFECT = (B(IAX,J,K)*(DOSEX*DOSEX))*FAC
```

5.0 SPAHR COMMON BLOCKS

SPAHR makes extensive use of common blocks to store data and transmit it between subroutines. The user who wishes to add commands to SPAHR will find it very useful to understand the structure and function of these blocks.

5.1 /TITLEC/

The common block /TITLEC/ contains information used to define page headings. Its use is described below.

```
                    C <TITLE> DATA BLOCK
        COMMON/TITLEC/IPAGE(20),LINES(20),TITLR(2),TITLJ(18)
```

IPAGE is a vector of page counts for each file. Every time the routine TITLE is called for logical unit I, the value in IPAGE(I) is incremented by 1.

LINES is a vector of line counts for the convenience of programs which must record the number of lines printed out on a given page. Whenever a call to TITLE or TITL is made, LINES(I) is reset to 0.

TITLR is not currently used.

TITLJ is a 72-character array that is printed out at the top of every page by TITLE or TITL.

5.2 /PUSHDC/

The block /PUSHDC/ stores the pushdown stack that is used to store the parameter block whenever a subcommand is invoked more than once.

```
C PUSHDOWN STACK
   COMMON/PUSHDC/NPUSH,KPUSH(60,5)
   DIMENSION RPUSH(60,5),JPUSH(60)
   EQUIVALENCE(KPUSH(1,1),RPUSH(1,1))
```

NPUSH is the total number of invocations currently called.

KPUSH is the array in which the parameter blocks are stored.

5.3 /CCARDC/

/CCARDC/ is the interface block for the free-format data entry and language processing routines.

```
C <FREEIO> INTERFACE BLOCK
   REAL*8 IWRD
   INTEGER*2 IC,KSPK,KDIG,KLET
   COMMON /CCARDC/ IWRD,INUM,NUMDIG,IEXP,I1,I2,I1S,IDLM,IER,NSPK,
       ITSO,ITERM,INTERM,IRES,IC(133),KSPK(12),KDIG(10),KLET(27)
```

IWRD	is an 8-byte string containing up to eight characters of the character string most recently read in by SCAN. IWRD may also be used to store the double-precision floating point version of the number most recently read in by RDNUM.
INUM	contains the integer most recently read in by RDINT.
NUMDIG	is the number of digits read in the most recent call to RDIG.
IEXP	is the value of the exponent last read in.
I1	is the pointer to the next character in the IC array to be processed in the current input record. I1 is reset by every routine that interprets any part of the current input record IC.
I2	is the pointer to the last defined character in the current input record. This is character 72 in the standard distribution version of SPAHR.
I1S	is the value of I1 prior to the initiation of a SCAN or numeric reference to the current line buffer. I1S is set to 1 by GETCRD.
IDLM	is the code for the delimiter following the last item read from the current input record.
IER	is used in conjunction with IDLM to infer the exact nature of the delimiting character. Some symbols, particularly the minus (-) and plus (+), have ambiguous meanings in some contexts. For the routines in which IDLM is set to zero, IER generally contains the negative of the IDLM value for - and +.
NSPK	is the number of special characters defined in the KSPK array.
ITSO	is a flag telling the processor whether the current execution mode is interactive (ITSO = -1) or batch (ITSO = 0).
ITERM	is the logical unit number for writing messages to the terminal or system log file.
INTERM	is the logical unit number for the command file.
IRES	is the index number for the terminal prompt message.
IC(133)	is the array containing the most recently read input record. For compatibility with the more primitive compilers, this is an array of 2-byte (16-bit) integers rather than of 1-byte logical (character) components. This is also true of the remaining arrays in this block.

KSPK(12) is an array of delimiting characters recognized by the language processor.

KDIG(10) is an array of character representations of the digits 0 through 9.

KLET(27) is an array of character representations of the letters A-Z. The 27th member is the underscore (_).

5.4 /CONSTC/

The block /CONSTC/ contains the constants of interest to most routines, such as age group definitions, sex labels, standard populations and their labels, and default values for many parameters.

```
C CONSTANT DATA BLOCK
    REAL*8 LAGE,LSEX
    COMMON/CONSTC/IOPTS(10),NA,NAM1,NAP1,NS,LAGE(20),LSEX(3),
   1 AGE(20),WGE(20),IAGE(20),STPP(20,2),STPPT,STPPX,NMSTPP(10,2)
    DIMENSION ROPTS(10)
    EQUIVALENCE (IOPTS(1),ROPTS(1))
```

IOPTS is an array used to hold default values and control parameters.

(1) is the timesharing status flag. It should be set to 0 for batch mode and -1 for timesharing mode. It is used to set the parameter ITSO in /CCARDC/.
(2) is the terminal or log file output unit number. It is used to set the /CCARDC/ parameter ITERM.
(3) is the assumed terminal linesize.
(4) is the default print file unit number.
(5) is the default detail file unit number (normally 0).
(6) is the default punch file unit number.
(7) is the number of age groups currently recognized (normally 19).
(8) is the number of sex groups (normally 2).
(9) is the batch mode switch.
(10) is the DO level beyond which page advisories are not printed out.

NA is the current number of age groups.

NAM1 is NA - 1.

NAP1 is NA + 1.

NS is the number of sex groups.

LAGE is the array of age group labels.

LSEX is the array of sex group labels.

AGE is the array of midpoints of the age groups in years.

WGE is the array of widths of the age groups in years.

IAGE is the array of starting ages for each age group in years.

STPP is the array containing the two standard populations.

NMSTPP is the array of 40-byte labels for the two standard populations.

5.5 /DATAC/

/DATAC/ is the current data block. It provides the storage for the demographic data for the population currently being processed, for all items except the cause-specific mortality data.

```
C CURRENT DATA BLOCK
      COMMON/DATAC/ITOC(28),IDATE,AOP,ASP,AKP,ACP,GFR,NRR,GRR,TFR,
     1    TP(3),TD(3),TB(3),ASDR(3,2),CDR(3),ASBR(3,2),CBR(3),
     2    SDR(3),SBR(3),GM(3),GOM(2,3),VLOG(3,3),NAME(18),LNM1,LNM2,
     3    FSC(20),FNM(20),FFX(20),BBT(20),BR(20),SEXR(20),PP(20,2),
     4    DD(20,2),BB(20,2),PPCT(20,2),VKX(20,2),SRX(20,2),DR(20,2),
     5    VL(20,2),VLL(20,2),VMX(20,2),QX(20,2),RX(20,2),AX(20,2),
     6    TX(20,2),EX(20,2),VDX(20,2),PI(100,2),DI(100,2),BI(100,2)
```

ITOC is the table of contents array for the current data block. The major arrays in the /DATAC/ block, as well as those in the /KOSC/ block, all have a member in this array. Each cell is set to 0 if the corresponding data item has never been set, to +1 if the data item was read in directly, and to -1 if the item was calculated within SPAHR. The association of each cell and data item will be described below.

IDATE is the year associated with the current population.

AOP is the mean age at childbirth for women in the current population.

ASP is the mean age of childbearing women in the stable population, corresponding to the current age-specific birth and death rates.

AKP is not currently used.

ACP is not currently used.

GFR is the general fertility rate, the total annual births divided by the number of women between 15 and 45 years of age.

NRR is net reproduction rate.

GRR	is gross reproduction rate.
TFR	is total fertility rate.
TP	is total population for females and males, both sexes combined.
TD	is the array of total deaths by sex.
TB	is the array of total births by sex.
ASDR	is the array of age-standarized death rates by sex for the two standard populations stored in SPAHR.
CDR	is the array of crude death rates by sex.
ASBR	is the array of age-standardized birth rates.
CBR	is the array of crude birth rates by sex.
SDR	is the array of death rates in stable population.
SBR	is the array of birth rates in stable population.
GM	is the array of geometric mean death rates. (Schoen's del.)
GOM	is the array of Gompertz parameters for the life table model.
VLOG	is the array of parameters for logistic life table model.

The remaining arrays are age-specific or are at least 20 words long. They are associated with an ITOC entry.

ITOC	Array	Meaning
1	NAME	Name to be associated with the current population.
2	FSC	Life table factor scores.
3	FNM	Net maternity function.
4	FFX	Top row of the Leslie matrix.
5	BBT	Total births by age of mother.
6	BR	Birth rates by age of mother.
7	SEXR	Sex ratios at birth by age of mother.
8	PP	Population by age and sex.
9	DD	Total deaths by age and sex.
10	BB	Total births by age of mother and sex of child.
11	PPCT	Proportional distribution of population by age for each sex.
12	VKX	Stable population proportional distribution.
13	SRX	Leslie matrix subdiagonals for each sex (projection ratios).
14	DR	Age-specific death rates for each sex.
15	VL	Life table survivors to beginning of age group.

16	VLL	Life table stationary age distribution.
17	VMX	Life table age-specific death rates.
18	QX	Life table probability of death in age group.
19	RX	Keyfitz local growth rate estimate.
20	AX	Life table number of years lived in each age group by those dying in it.
21	TX	Life table T column.
22	EX	Expectation of life at beginning of each age group.

5.6 /KOSC/

/KOSC/ is the second major data block in SPAHR. It contains data relating to mortality by cause.

```
C CAUSE-OF-DEATH DATA BLOCK
      REAL*8 LKOS,KOSID
      COMMON/KOSC/NKOS,NKM,KOS1,KOS2,LKOS(36),TDC(36,2),
     1 DRC(20,2,36),FSCC(5,36),KOSID(9,36),XMRB(8,2,36),IMD(36)
      DIMENSION DATKOS(1348), DDC(20,2,36)
      EQUIVALENCE (NKOS,DATKOS(1)),(DDC(1,1,1),DRC(1,1,1))
```

NKOS is number of causes of death currently in use.

NKM is number of causes of death permitted.

KOS1 is not currently used.

KOS2 is not currently used.

LKOS is the array of labels used to identify the defined causes.

TDC is the array of total deaths by cause and sex.

DRC is the array of age specific death rates by sex and cause, or of deaths by age and cause. Deaths are indicated when ITOC(23) is set to 0 and ITOC(24) is set to +1 or -1; rates are indicated if the reverse is true.

FSCC is the array of principal component factor scores for the age pattern for each cause.

KOSID is an array of 72-character labels for each cause of death, used for certain types of graphic output.

5.7 /PROJC/

The block /PROJC/ contains the control variables for the PROJECT command.

```
C <PROJECT> PARAMETER CONTROL BLOCK
      REAL*8 LTYP
      LOGICAL LFERT,LMORT,LADJ,LADJA,LADJC,LADJM,LADJF
      COMMON/PROJC/IPPR(40),IEAR,IEARP5,IPUSH,LFERT,LMORT,LADJ,LADJA,
      ADJDA,
     1 LADJC,LADJM,LADJF,ISTBR,ISTPBR,ISTDR,ISTPDR,LTYP,NREPT,NRPT,
     2 NKSAVE,NPSAVE,NKOSQ,KOSO,INRPT,
     3 IREPT,IEARO,XDIF1,XDIF2,KO,IFDS,NMSTOR(18)
       DIMENSION IFRX(62),RPPR(40)
       EQUIVALENCE(IPRX(1),IPPR(1),RPPR(1))
```

IPPR is the array of control parameters that can be set with a call to GETPAR. The members are defined as follows:

(1) NA	(16) STARTDR
(2) not used	(17) STOPDR
(3) START	(18) STARTBR
(4) STOP	(19) STOPBR
(5) DETAIL	(20) DOSINT
(6) POPSIZE	(21) COHORT
(7) not used	(22) PERIOD
(8) not used	(23) NCOLS
(9) ACCUM	(24) RATES
(10) PYRAMID	(25) C2
(11) PRINT	(26) CONSTANT
(12) PUNCH	(27) PRJSRV
(13) INCREM	(23) ADJDUR
(14) RETAINDR	(29) PRJKOS
(15) RETAINP	(30) RANDOM

IEAR is the initial year of the current projection interval.

IEARP5 is the terminal year of the current projection interval.

IPUSH is the current level in the subcommand pushdown stack.

LFERT is reserved for future use.

LMORT is reserved for future use.

LADJ is set to .TRUE. if any of the rate-adjusting subcommands were set while the command was being interpreted (i.e. BIRTHL, DEATHL, or ADJUST).

ADJDA is set to .TRUE. if during the current projection interval the routine ADJDUR was called and if it made alterations in any of the vital rates.

LADJC	is set to .FALSE. if a period projection is in progress and to .TRUE. if a cohort projection is in progress. LADJC is set on entry by PRJPER and PRJCOH.
LADJM	is set to .TRUE. if the death rates were changed by interpolation during the current projection interval. It is reset to .FALSE. at the end of each projection interval, because it is used to determine whether to call PRJSRV during each interval.
LADJF	is set to .TRUE. if the birth rates were altered by interpolation during the current interval and is set to .FALSE. otherwise.
ISTBR	is the initial interpolation year for birth rates.
ISTPBR	is the final interpolation year for birth rates.
ISTDR	is the initial interpolation year for death rates.
ISTPDR	is the final interpolation year for death rates.
LTYP	is an eight-character string defining the mode of the current projection. It is used primarily for labeling purposes. LTYP is set to BACKGROU prior to the background mortality adjustment call to ADJDUR from PRJECT, to PERIOD from PRJPER, and to COHORT from PRJCOH.
NREPT	is the upper bound for the number of repetitions (see IREPT below). NREPT has a value > 1 ONLY if RANDOM mode is in effect.
NRPT	is the number of short records written out for each projection on unit 1 in either RANDOM or ACCUM mode. NRPT tells a POSTPROC mode projection how many short records to expect before the final (long) record.
NKSAVE	is the number of causes to be passed to a POSTPROC mode projection.
NPSAVE	is the value of NPUSH passed to a POSTPROC mode projection.
NKOSQ	is the number of causes defined prior to the current projection. NKOSQ is used to reset NKOS at termination unless RETAINDR was specified.
KOSO	is the index number of the _ORIGNL_ cause group if one was created in the current projection. KOSO is only used if one or more ADJUST calls were made.
IREPT	is the repetition loop control variable. It is set in PRJPER to measure the number of times the current simulation has been repeated. Unless the simulation is in RANDOM mode, IREPT will have the value 1 after PRJPER has begun. Thus, routines that take special action during Monte Carlo simulations check the value of IREPT to determine in which mode they are operating.

IEARO is the value IEAR had at the first projection interval in the current print storage block.

XDIF1 is set to the value of STOPDR - STARTDR. It is used as the denominator for interpolation calculations to avoid unnecessary computations.

XDIF2 is set to the value of STOPBR - STARTBR.

KO is the value of the K index in the print storage arrays at the time when it was last dumped.
Note: The K index is incremented for each projection interval and is reset to 1 when the storage arrays are printed out.

IFDS is the file unit in which the output generated by the DOSINT option is placed. IFDS appears in the control block because the output varies depending on whether IFDS is set equal to ITERM in /CCARDC/.

NMSTOR is a 72-character array in which the NAME string from /DATAC/ is placed at the beginning of the PROJECT command execution and from which NAME is restored at the end if any alterations were performed on NAME during the projection. The NAME array is usually altered when printing out results from cohort and Monte Carlo simulations.

IPRX is equivalenced to the beginning of the block. IPRX is used to restore parts of the block when operating in RANDOM mode.

5.8 <u>/ / in PROJECT</u>

The blank common block is used by the PROJECT command to store a variety of arrays containing summaries of projection results (often referred to as <u>print control arrays</u>). It is also used to store initial population, current vital rates, and rates to be used for interpolations.

```
C C BLANK COMMON : COMMAND <PROJECT>
     COMMON BRT(2,200),VLX(20,2),
    1 VLLX(20,2),PSTRT(20,2),FF(20),DRAT(20,2),SR(20,2),RV(20,2),
    2 DRX(20,2),DRY(20,2),DRZ(20,2),DRL(20,2),BRX(20),BRZ(20),
    3 BRL(20),ITYR(2,11),POP(20,2,11),DTH(20,2,11),BTH(2,11),
    4 POPTOT(11),DCT(2,36,11),VSTAT(12,2,11),FRI(3,11),BBA(20,2,11),
    5 PYRS(20,3),DDSUM(20,3),PYMORB(20,2,3),BRSUM(3),DRSUM(3),
    6 BBSUM(3),GROWTH(3),GRSUM(3),DCBOTH(36),DOSINT(36),ESTIM(36),
    7 DCSUM(2,36),XINCI(2,36),SMORB(2,36),PYRSO(3),
    8 POPOM(20,2,11),POPMO(20,2,20),BTHMO(2),DTHMO(20,2),
    9 BTHOM(2,11),DTHOM(20,2,11),EMIGR(20,2),MOYEAR(20) C C
 EQUIVALENCES FOR SPECIAL PURPOSES
      DIMENSION RPPR1(20),IPPR1(20),RECO(2442),REC1(3024)
      EQUIVALENCE(DRZ(1,1),RPPR1(1),IPPR1(1)),
    1  (RECO(1),REC1(1),POP(1,1,1))
```

BRT	is the birth trajectory array. As each period in the period projection is generated, the births for females and males are placed in BRT(1,*) and BRT(2,*), respectively. The BRT array is then used by PRJCOH as the starting point for projecting cohorts born after the START date of the projection.
VLX	is the life table survivors array generated in the last call to PRJSRV.
VLLX	is the life table person-years array generated in the last call to PRJSRV.
PSTRT	is the initial population to be projected, which is used as a starting point by both PRJSRV and PRJCOH for the current population.
FF	is the top row of the Leslie matrix generated by the last call to PRJSRV.
DRAT	is the array of death allocation ratios calculated in the last call to PRJSRV.
SR	is the array of survival ratios calculated in the last call to PRJSRV.
RV	is the array of other items from PRJSRV. For details about its contents, see the section about PRJSRV, Chapter 4.3.6.
DRX	is the array of death rates applicable to the current projection interval.
DRY	is the array that retains a copy of DRX for use when the value of DRX depends upon itself, as in some calls to ADJDUR. NOTE: DRY is redundant in most of its uses, and will probably be eliminated or rededicated in future releases. The user should not become dependent upon it.
DRZ	is the array of initial death rate values for interpolation.
DRL	is the array of final death rate values for interpolation.
BRX	is the array of current birth rates by age of mother.
BRZ	is the array of initial birth rates for interpolation.
BRL	is the array of final birth rates for interpolation.
ITYR	is the array of year labels for printing column titles in the summary output tables. For each value of K, ITYR(1,K) contains the value of IEAR then prevailing, and ITYR(2,K) contains MOD(IEARP5,5)-1.

POP	is the array of projected populations by age and sex at the beginning of each projection interval.
DTH	is the array of deaths by age during each projection interval.
BTH	is the array of births by sex for each projection interval.
POPTOT	is the array of total populations at the beginning of each projection interval.
DCT	is an array of deaths by cause and sex for each projection interval.
VSTAT	is an array of summary statistics describing various aspects of the population during each projection interval. For details see the description of PRJPER, Chapter 4.3.2.
FRI	is temporary storage for the NRR, GRR, and TFR calculated for each projection interval.
PYRS	is the array of a running accumulation of all person-years lived by age and sex since the projection began.
DDSUM	is the array of a running accumulation of all deaths by age and sex since the projection began.
PYMORB	is an array set in PRJMC1. PYMORB holds the estimate of person-years lived in various states of impaired health during the course of the projection.
BRSUM	is an array calculated at the end of the projection to contain the overall birth rate for each sex.
DRSUM	is calculated at the end of the projection to contain the overall crude death rate for each sex.
BBSUM	is an array of a running accumulation of all births by sex.
GROWTH	is set at the end of the projection to be the net change in population size over the course of the projection for each sex.
GRSUM	is the overall growth rate for each sex during the course of the projection.
DCBOTH	is an array of the running sum of deaths by cause, both sexes combined.
DOSINT	is an array of the running accumulation of integrated exposure to the hazard of interest applicable to each cause of death.

ESTIM is an array of the risk estimators for the projection for each cause, calculated in routine INTDOS and finally in PRJPER as

ESTIM = DCBOTH / DOSINT.

DCSUM is an array of the running accumulation of deaths by cause by sex.

XINCI is an array of the estimated incidence of disease by cause based on the cause-specific death rate.

SMORB is an array of the estimated number of person-years elapsing between diagnosis and death or recovery for each cause.

5.9 /ADJSTC/

The block /ADJSTC/ is the control and parameter block for the ADJUST subcommand of the PROJECT command. This block is structured as if it were to be initialized only once in routine ADJUST and used thereafter. However, the routines that use it may be invoked several times. Therefore, each time routine ADJUST is called, it saves the first 45 words in /PUSHDC/. Later, when ADJDUR is called to use the data, it passes through /PUSHDC/ the number of times indicated and resets the first 45 words.

```
C C DEATH RATE ADJUSTMENT DATA BLOCK
      DIMENSION MINAGE(2),MAXAGE(2),CRT(9,2,28),RPR(30)
      COMMON/ADJSTC/IPR(30),DURP,NK,KS1,KS2,NDS1,NDS2,PEX,ICALL,
     1  IR,MODTYP,IVARY,NCF1,NCF[illegible]NGR1,NGR2,IDOSL,CC(18,28),
     2  LATNT(9,28),LPLT(9,28)[illegible],LDOSE,NDM,IDOSE(28),
     3  DOSE(251,28)
      EQUIVALENCE (IPR(1),RPR(1[illegible],CRT(1,1,1)),
     1  (MINAGE(1),IPR(21)),(MAX[illegible],IPR(23))
```

RPR is the single-valued parameter array. Its members are

(1) PRINT	(15) LPLAT
(2) DETAIL	(16) START
(3) MODEL	(17) STOP
(4) CIG	(18) REM
(5) POM	(19) BACKGROU
(6) NOX	(20) INPUT
(7) SO2	(21) MINAGEF
(8) SO4	(22) MINAGEM
(9) TSP	(23) MAXAGEF
(10) RFRAC	(24) MAXAGEM
(11) ERESP	
(12) ENRESP	
(13) ETOT	
(14) REL	

DURP	is the difference between the START year of the projection and the START year of the current invocation of the ADJUST subcommand. DURP is used to determine the relative position of the beginning entry in the DOSE array and the current year.
NK	is the number of causes of death recognized in the current ADJUST call.
KS1	is the index number of the first cause group in the DRC array in /KOSC/ to be used for relative risk computations in the current model.
KS2	is the index number of the first cause of death in the DRC array in /KOSC/ defined by the current call to ADJUST.
NDS1	is the index of the first dose vector in the DOSE array applicable to the current ADJUST call.
NDS2	is the index number of the last dose vector in the DOSE array applicable to the current ADJUST call.
PEX	is the constant exposure level for the current pollutant (see IVARY below).
ICALL	is a counter for the number of times the adjustments defined in the current ADJUST call have been made.
IR	is the relative risk mode switch. =0 indicates absolute risk mode. =1 indicates relative risk mode.
MODTYP	is the model type index. =1 indicates an air pollution model. =2 indicates a radiation model.
IVARY	is the variable exposure history index. =0 indicates that a constant exposure value is stored in PEX. ≠0 indicates that a variable exposure history is stored in DOSE.
NCF1	points to the first coefficient set (i.e. it contains the value of the second index variable) applicable to the current invocation level of ADJUST. NCF1 is used in reference to the arrays CC, LATNT, LPLT, and IPRK.
NCF2	points to the last coefficient set in the current model.
NGR1	points to the first group name in the current model. The group names are stored in variable LKOS in /KOSC/.

NGR2	points to the last group name in the current model.
IDOSL	is the maximum number of years actually used in the largest vector in DOSE under the current ADJUST invocation.
CC	is the array holding the primary coefficients in the current model. Unlike the other coefficient arrays below, CC is sex specific, and is therefore equivalenced to CRT to facilitate sex-specific computations.
LATNT	is an auxilliary coefficient array used in some models to define cause-specific waiting periods during which no effects are observed.
LPLT	is an auxilliary coefficient array used in some models to define the length of time following the LATNT period during which effects can be observed.
IPRK	points to the cause of death in the DRC array of /KOSC/ that has the same name as the defined cause in the current model, if present.
LDOSE	is the maximum number of years available in the variable DOSE array.
NDM	is the number of dose history vectors available in the DOSE array. (DOSE is assumed to have the de facto declaration DOSE(LDOSE,NDM).)
IDOSE	is an array containing the real lengths of the vectors in DOSE.
DOSE	is the array of variable dose histories. The units are determined by the nature of the model under which they are stored.

6.0 IMPLEMENTING SPAHR ON THE COMPUTER

This chapter covers the implementation and running of SPAHR on a computer system. As of this writing, SPAHR has been run only on IBM computers in the 360 series (360/75, 370/168, 370/195, and 3033). It has been tested extensively under the OS/MVT and CMS operating systems. Because SPAHR is written entirely in IBM FORTRAN, it may in principle be run on any other computer running a compatible compiler. These include most byte-oriented machines that recognize the *n extension of the type statements.

The procedure for setting up SPAHR on the host computer is straightforward. The source code and, in an IBM environment, the load module are copied from the distribution tape. Then either the source code is recompiled and an executable form of the program prepared, or the load module is directly set up in a library. Several sets of test data and sample problems are included on the distribution tape to facilitate checkout of the program.

6.1 SPAHR Distribution Tape

SPAHR is distributed on a 1600 bpi tape containing the documentation, IBM load module, source code, and several sample jobs and sets of test data. Except for the load module file (which is always the second file on the tape) all files are blocked files with fixed-length, 80-byte records, 39 records per block (for a blocksize of 3120 bytes). All files except the second may be copied by using the IBM utility program IEBGENER. The second file, which contains the load module, is a partitioned data set, and has been loaded onto the tape with the IBM utility program IEBCOPY. It must therefore be unloaded using IEBCOPY. The other popular utility used on partitioned data sets, IEHMOVE, uses a different tape format and therefore cannot be used on the second file.

Unless otherwise specified, the sample command files TEST1 through TEST9 on the SPAHR tape assume that the sample data file DATA1 has been assigned to unit 4, and that the sample data file DATA4 has been assigned to unit 11. Furthermore, unless otherwise specified, the SPAHR files in DATA1 are all documented internally, except for file TEST1, which is documented in Chapter 4 of the User's Guide (Volume IV).

The SPAHR tape has the following structure:

File	Data Set Name	Contents
1	GUIDE	Formatted copy of the current version of the SPAHR User's Guide.
2	LOAD	Load module for SPAHR.

3	FORTRAN1	First source code file. This file contains all of the language processor routines used in SPAHR.
4	FORTRAN2	Contains all small utility routines shared by the various command drivers in SPAHR. It also contains the OPTIONS command driver.
5	FORTRAN3	Source code for all command processors except OPTIONS, PROJECT, and FACTOR.
6	FORTRAN4	Source code for the PROJECT command processor.
7	FORTRAN5	Source code for the FACTOR command processor.
8	TEST1	Contains all of the sample command files from Chapter 4 of the User's Guide.
9	TEST2	Sample job illustrating free-format input data from the command file and a variety of procedures including population projections with varying birth and death rates.
10	TEST3	Sample job illustrating free-format input data from an auxiliary file on unit 10 (the data in question are assumed to come from file DATA1 on this tape). Two examples of radiation risk projections are shown. The first illustrates integrated exposure calculations, while the second shows life table calculations.
11	TEST4	Illustrates the use of multiple ADJUST invocations in the PROJECT command, and the use of the RETAINx parameters.
12	TEST5	Cohort projection demonstrations.
13	TEST6	Illustrates the use of SPAHR DO loops, PRINT statements, variables, and a combination of fixed-and free-format input.
14	TEST7	Illustrates a Monte Carlo population projection scheme using the RANDOM parameter in PROJECT.
15	TEST8	Illustrates accumulated and pooled projections.
16	TEST9	Cause-of-death analysis using the cause groups defined by Preston, Keyfitz, and Schoen (1972) for the United States. The first set of data is entered from the command file. The remaining data are read using standard format 3. The data from file DATA5 on this tape are assumed.

17	DATA1	Free-format data file for the U. S. White population in 1970 with population, birth rates, total deaths, and deaths by cause for the radiogenic cancer categories established in the Rasmussen report (USNRC, 1976).
18	DATA2	Similar to DATA1 for the U. S. Black population in 1970.
19	DATA3	The data of files DATA1 and DATA2, but in standard format 1 (the SPAHR standard format).
20	DATA4	Data in standard format 2 (the Keyfitz standard) for 31 selected countries.
21	DATA5	Selected data in standard format 2 from the Preston, Keyfitz, and Schoen data base, covering the United States from 1920 to 1970.

6.2 Reading the Distribution Tape

The distribution tape contains 21 files. Copying these onto disk files is the first step in implementing SPAHR on your local computer system. The procedure shown below should perform the copying at an IBM computer installation running under OS. A copy of the cards listed below is included with the SPAHR tape. Only those lines containing a definition enclosed in angle brackets (<>) need to be modified before running. The <prefix> must terminate with a period, and must be enclosed in quotation marks in the exec statements.

At some installations, cataloging datasets is discouraged. Instead, a specific disk must be selected. At these installations, the sections of the code below that read DISP=(NEW,CATLG) should be changed to read DISP=(NEW,KEEP), and the VOLUME parameter should be coded appropriately.

```
//*
//* UTILITY PROCEDURE FOR COPYING TAPE
//*
//COPYTAPE PROC DSN=,FILE=,UNIT=<tape_unit>,PREFIX='<prefix>'
//COPY EXEC PGM=IEBGENER
//SYSPRINT DDD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DISP=(OLD,PASS),VOL=(,RETAIN,,SER=(<tape_#>)),
//  LABEL=(&FILE,SL),UNIT=&UNIT,DSN=&DSN
//SYSUT2 DD DISP=(NEW,CATLG),SPACE=(TRK,(19,10),RLSE),
//  UNIT=<disk_class>,DSN=&PREFIX.&DSN
//PEND
//*
//* SCRATCH ANY EXISTING VERSIONS OF THESE FILES
```

```
//*
//SCRATCH PROC PREFIX='<prefix>'
//PURGE EXEC PGM=IEFBR14
//LD DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.LOAD
//S1 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.FORTRAN1
//S2 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.FORTRAN2
//S3 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.FORTRAN3
//S4 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.FORTRAN4
//S5 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.FORTRAN5
//T1 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST1
//T2 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST2
//T3 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST3
//T4 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST4
//T5 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST5
//T6 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST6
//T7 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST7
//T8 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST8
//T9 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.TEST9
//D1 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.DATA1
//D2 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.DATA2
//D3 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.DATA3
//D4 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.DATA4
//D5 DD DISP=(MOD,DELETE),UNIT=DISK,DSN=&PREFIX.DATA5
// PEND
//*
// EXEC SCRATCH
//*
//* PRINTS OUT A COPY OF THE DOCUMENTATION
//*
//DOCUMENT EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DISP=(OLD,PASS),UNIT=<tape_unit>,
//  DSN=GUIDE,VOL=(,RETAIN,,SER=(<tape_#>)),LABEL=(1,SL),
//SYSUT2 DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=80,BLKSIZE=1600)
//*
//* OS LOAD MODULE FOR SPAHR
//*
//LOADMOD EXEC PGM=IEBCOPY
//SYSIN DD DUMMY
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1573)
//SYSUT3 DD UNIT=SASCR,SPACE=(CYL,(1,1))
//SYSUT2 DD DISP=(NEW,CATLG),UNIT=<disk_class>,
//  SPACE=(CYL,(4,1,5),RLSE),DSN=<prefix>LOAD
//SYSUT1 DD DISP=(OLD,PASS),UNIT=<tape_unit>,
// LABEL=(2,SL),DSN=LOAD,VOL=(,RETAIN,,SER=(<tape_#>))
//*
//* SPAHR SOURCE CODE
```

```
//*
//SOURCE1 EXEC COPYTAPE,DSN=FORTRAN1,FILE=3
//SOURCE2 EXEC COPYTAPE,DSN=FORTRAN2,FILE=4
//SOURCE3 EXEC COPYTAPE,DSN=FORTRAN3,FILE=5
//SOURCE4 EXEC COPYTAPE,DSN=FORTRAN4,FILE=6
//SOURCE5 EXEC COPYTAPE,DSN=FORTRAN5,FILE=7
//*
//* TEST JOB FILES
//*
//TESTD1 EXEC COPYTAPE,DSN=TEST1,FILE=8
//TESTD2 EXEC COPYTAPE,DSN=TEST2,FILE=9
//TESTD3 EXEC COPYTAPE,DSN=TEST3,FILE=10
//TESTD4 EXEC COPYTAPE,DSN=TEST4,FILE=11
//TESTD5 EXEC COPYTAPE,DSN=TEST5,FILE=12
//TESTD6 EXEC COPYTAPE,DSN=TEST6,FILE=13
//TESTD7 EXEC COPYTAPE,DSN=TEST7,FILE=14
//TESTD8 EXEC COPYTAPE,DSN=TEST8,FILE=15
//TESTD9 EXEC COPYTAPE,DSN=TEST9,FILE=16
//*
//* TEST DATA SETS
//*
//DATA1 EXEC COPYTAPE,DSN=DATA1,FILE=17
//DATA2 EXEC COPYTAPE,DSN=DATA2,FILE=18
//DATA3 EXEC COPYTAPE,DSN=DATA3,FILE=19
//DATA4 EXEC COPYTAPE,DSN=DATA4,FILE=20
//DATA5 EXEC COPYTAPE,DSN=DATA5,FILE=21
```

6.3 SPAHR Files and Job Control

All computer programs communicate with their users through the use of files, and SPAHR is no exception. A computer file is an organized block of information on some machine-readable medium such as disk, magnetic tape, or terminal. SPAHR files may be classified as follows:

1) The command file contains all SPAHR command statements. It may also contain input data for the DATA command. In batch execution, the command file is an ordinary card-image file. In interactive mode, the command file is the terminal input. The command file is always on file unit 5.

2) The log file is a file on which SPAHR writes out brief summaries of its activities and error messages. In batch mode, it is an ordinary print file. In interactive mode, it is the terminal output. The log file uses file unit number 6.

3) A print file holds the ordinary output from the various SPAHR procedures. In both batch and interactive mode, output to a print file is sent to a line printer. The default unit for printed output is unit 3. More than one print file may be used simultaneously.

4) Output destined to be read in by another computer program is put on a card-image file. The keyword PUNCH is used in most cases to control the production of such data files. The default unit for card-image files is 7. More than one card-image file may be used during a SPAHR run.

5) A data file contains fixed-format or free-format data to be read in to SPAHR. The data file is assumed by SPAHR to be in the form of a card-image file. By default, unit 4 is assumed whenever a data file is needed unless SPAHR is told otherwise. More than one data file may be specified during a run.

6) Scratch files are files used for the temporary storage of information in free form. SPAHR uses only one scratch file at any one time, on file unit 1.

SPAHR files, particularly the command and data files, are discussed in more detail and with examples in the SPAHR Introductory Guide, Volume II of this report.

Before SPAHR is used on the host computer, its operating system must define the files to be used and make them available to the program. The process of directing the computer to execute the program is called job control. The sections below (6.3.1 and 6.3.2) will show how to direct various operating systems. More information on this subject is given in the SPAHR User's Guide, Volume IV.

6.3.1 OS Batch Execution - Cataloged Procedure

At most IBM installations that run a version of the OS operating system, the job control language (JCL) required to execute a program such as SPAHR can be placed in a cataloged procedure, which can then be referenced by the user with a one-or two-line EXEC statement in place of the voluminous mass of instructions that would otherwise be required. The JCL cards described here may be used directly in a job as well.

The block sizes shown for the print and scratch files in this procedure are those recommended for installations running under the ASP 3.3 system using 3330-style disk drives. Other block sizes may be more suitable for different installations.

OS Example 1: Recommended Cataloged Procedure

```
//*************************************************************************
//*
//*                         *** S P A H R ***
//*
//*          THIS PROCEDURE EXECUTES THE SPAHR RISK PROJECTION
//*           AND DEMOGRAPHIC ANALYSIS PROGRAM.  TO USE, CODE
//*
//*                 //  EXEC  SPAHR,DATA='dsn_of_data_file'
//*                 //SYSIN DD *
//*                   ... Spahr command file ...
//*
//*************************************************************************
//SPAHR    PROC   DATA=,  DSNAME OF UNIT 4 DATA FILE
//                UNIT=,  UNIT NAME FOR DATA FILE
//                VOL=    VOLUME SERIAL NO. FOR DATA FILE.
//*
//GO       EXEC PGM=SPAHR
//STEPLIB  DD DISP=SHR,DSN=<dsname_of_load_module>
//FT01F001 DD DISP=(NEW,DELETE),UNIT=DISK,         DEFAULT
//         SPACE=(CYL,(2,2)),                          SCRATCH
//         DCB=(RECFM=VBS,RLRECL=X,BLKSIZE=3120)          FILE
//FT03F001 DD SYSOUT=A,                            DEFAULT
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1596)    PRINT FILE
//FT04F001 DD DISP=SHR,DSN=&DATA,                  DEFAULT
//         UNIT=&UNIT,VOL=SER=&VOL                     DATA FILE
//FT06F001 DD SYSOUT=A,                            DEFAULT
//         DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1596)    LOG FILE
//FT07F001 DD SYSOUT=B,                            DEFAULT
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)        PUNCH FILE
//FT05F001 DD DDNAME=SYSIN
```

6.3.2 CMS Execution - EXEC File

Executing SPAHR under the CMS system also requires job control directives. Like OS batch, these directives may be isolated in a separate file, called an EXEC file, that can be referenced as a single command.

The CMS system has a much more flexible job control system than does OS. However, with this flexibility comes a disadvantage: CMS exec files can be much longer and more complicated than their counterparts in the OS cataloged procedure. The OS cataloged procedure described in Chapter 6.3.1 merely substitutes certain arguments into an otherwise fixed job control structure. The job control processor under CMS, however, is a full-blown interpretive language processor that permits almost unlimited possibilities for looping, conditional execution, defining defaults, and revising command syntax. Therefore an example of a SPAHR EXEC file will be given. Except for the illustration denoted by ?, the example given for CMS in Chapter 5.2 of Volume IV, the SPAHR User's Guide, will work with the sample EXEC file given here.

This sample EXEC file is divided for purposes of discussion into three logically distinct segments. The first segment consists for the most part of the online documentation and the logic used in deciding whether to print the file. The second section parses the command line (if this is not simply a documentation call) and allocates default values to all necessary parameters. The third section is similar in function to the primitive exec: it allocates all necessary files based upon the parameters defined in the second segment, executes the SPAHR program module, and then disposes of files as needed after execution is complete.

CMS Example

```
*******
*
*    S P A H R
*
******* IS THIS A DOCUMENTATION CALL?
&IF .&1 =    .? &GOTO -TELL
&IF .&1 ¬= .AUTHOR &GOTO -PARSE
&TYPE Written by R. T. LUNDY    BIM    B24329 ext. 2-3827
&EXIT 0
-TELL & BEGTYPE ALL
SPAHR fn1 fn2 ( PRINT xxx LOG yyy)

 where fn1      is the file name of /fn1 COMMAND */, which is the SPAHR command
                file.  If an astrick is entered, command lines will be taken
                directly from the terminal.

       fn2      is the file name of /fn2 DATA */, which is a SPAHR data file
                that will be allocated to unit 4.  By default you get /DATA1
                DATA */.

       xxx      (default DISK) is the disposition of the SPAHR print file
                allocated to unit 3.  It may be one of:

                DISK  - Directs that the print file be placed on disk file
                        /SPAHR3 LISTING E/.

                PRINT - Directs that the print file be allocated to the user's
                        CMS virtual printer.

       yyy      (default TERM) is the disposition of the SPAHR log file
                allocated to unit 6.  It may be one of:

                TERM  - Directs that the log file be printed out at the user's
                        terminal.
```

```
               DISK  - Directs that the log file be placed on disk file
                       /SPAHR6 LISTING A/.

               PRINT - Directs that the log file be allocated to the user's
                       CMS virtual printer.

&END
&EXIT 0
*******
*
* THIS SECTION PARSES THE COMMAND LINE AND ASSIGNS DEFAULTS.
*
*******
-PARSE
*
******* DEFINES ERROR RESPONSE
&CONTROL ERROR
&ERROR &EXIT -101
*
******* SET DEFAULT VALUES
&DATFILE = DATA1
&COMFILE = *
&PRDISP  = DISK
&LOGDISP = TERM
&PRLOG   = 0
*
******* PARSE THE COMMAND LINE
&N = 0
*
******* POSITIONAL PARAMETER SECTION *******
-NEXT &N = &N + 1
&IF &N > &INDEX &GOTO -RUN
&IF &&N = ( &GOTO -NEXTP)
&IF &N = 1 &GOTO -COM
&IF &N = 2 &GOTO -DAT
&GOTO -NEXTP
-COM &IF .&1 = . &GOTO -NEXT
&IF  &1 = * &GOTO -NEXT
&COMFILE = &1
&GOTO -NEXT
-DAT &IF .&2 = . &GOTO -NEXT
&DATAFILE = &2
&GOTO -NEXT
*
******* KEYWORD PARAMETER SECTION *******
-NEXTP &N = &N + 1
&IF &N > &INDEX &GOTO -RUN
```

```
******* JUMP TO THE SECTION FOR THE CURRENT KEYWORD
&GOTO -&&N
******* KEYWORD /PRINT/
-PRINT &N = &N + 1
&PRDISP = &&N
&GOTO -NEXTP
******* KEYWORD /LOG/
-LOG &N = &N + 1
&LOGDISP = &N
&GOTO -NEXTP
*******
*
* THIS SECTION INITIALIZES FILES AND EXECUTES THE SPAHR MODULE
*
*******
*
******* ALLOCATE PRINT FILE TO A PRINTER UNLESS /DISK/ IS SPECIFIED
-RUN &IF &PRDISP ¬= DISK &GOTO -RUN1
FILEDEF 3 DISK SPAHR3 LISTING E (RECFM FBA   LREDL 133 BLKSIZE 1596)
&GOTO -RUN2
-RUN1 FILEDEF 3 PRINT <RECFM FBA LRECL 133  BLKSIZE 1596
*
******* ALLOCATE LOG FILE
-RUN2 &IF &LOGDISP = TERM &GOTO -RUN3
&IF &LOGDISP ¬= DISK & PRLOG = 1
FILEDEF 6 DISK SPAHR6 LISTING E <RECFM FBA LRECL 133
&GOTO -RUN4
-RUN3 FILEDEF 6 TERM <LRECL 133
*
******* COMMAND FILE
-RUN4 &IF &COMFILE ¬= * &GOTO -RUN5
FILEDEF 5 TERM
&GOTO -RUN6
-RUN5 STATE &COMFILE COMMAND *
FILEDEF 5 DISK &COMFILE COMMAND *
*
******* DATAFILE
-RUNS STATE &DATFILE DATA *
FILEDEF 4 DISK &DATFILE DATA *
*
******* PUNCH FILE
FILEDEF 7 DISK SPAHR7 DATA A
*
******* SCRATCH FILES
FILEDEF 1 DISK SPAHR1 SCRATCH A
FILEDEF 2 DISK SPAHR2 SCRATCH A
*
```

```
******* RESPOOL PRINTER IF LOG FILE IS TO BE ATTACHED
&IF &PRLOG = 1 CP SPOOL PRINT CONT
*
******* EXECUTES THE FILE /SPAHR MODULE/
*
SPAHR
*
******* PRINT OUT THE LOG FILE IF /LOG PRINT/ WAS SPECIFIED
*
&IF &PRLOG = 0 &EXIT
PRINT SPAHR6 LISTING E
CP SPOOL PRINT CLOSE NOCONT
&EXIT 0
COMMAND? .
```

6.4 Maintaining and Modifying the SPAHR Code

In order to add features to the SPAHR system, or to repair deficiencies that may be discovered in the existing code, the user may desire to modify the operation of selected parts of the SPAHR program. At this point the disadvantages of a large program package become most apparent. Recompiling and reloading the entire code is a nontrivial operation that exceeds the limits applied to ordinary users at most computer centers. It is preferable to recompile only those routines that require modification and reconstruct the load module (the executable form of the program). The mechanics of this operation will depend on the characteristics of the user's particular installation.

One rule should be observed at all times: Never alter an existing dataset until its replacement has been created and checked out !! There are two reasons for following such a policy. First, the user retains the option of returning to the original version of the program if the modification is less satisfactory than the original version. Second, the user minimizes the risk that a failure in the computer's operating system will destroy the load libraries or datasets while they are being changed. System failures of this nature are not at all uncommon. While the procedure recommended below may appear to be somewhat overburdened with redundant steps and crosschecks, experience suggests that the extra effort will be worthwhile.

6.5 Maintenance Procedures at Argonne National Laboratory: CMS

The SPAHR system is maintained on a CMS virtual disk on the ANL CMS system. It is available in three forms:

1) FORTRAN source decks. Each subroutine is available as a file with filetype FORTRAN. The major common blocks are replaced with a record of the form

```
-INC blockname
```

starting in column 1. One of the utility routines described below expands the -INC cards into the full source for the common block by using the file blockname SCRIPT, which is also stored on the SPAHR disk. The filetype SCRIPT is used to simplify the interface with the processor that generated the internal documentation.

2) Object code decks. For every FORTRAN deck, there is a corresponding file of type TEXT containing the compiled code for the subroutine. These TEXT Files are needed whenever a recompilation and test load are to be performed, or whenever a new MODULE file is to be created. (Unlike the OS system, the existing load module cannot be relinked with new subroutines.)

3) A MACLIB source library. MACLIB contains a member corresponding to each of the subroutines in expanded source code form. (I.e., the -INC cards have all been replaced by the corresponding COMMON blocks.) MACLIB is required whenever the WATFIV debugging compiler is used.

4) A file named SPAHR MODULE contains the executable form of the program.

The documentation for the SPAHR program is also maintained on the SPAHR disk in the format designed for the SCRIPT text formatting program. Documentation files all have the filetype script. Sections of the SPAHR User's Guide (Volume IV) have filenames that begin with GUIDEU. Sections of the SPAHR Programmer's Guide (Volume V) have filenames that begin with GUIDEP. Additional files of type SCRIPT may be imbedded in these files. Some of these, such as file OSJCLANL SCRIPT, are permanently in residence on the SPAHR disk. Others must be created whenever the document is printed off by copying COMMAND or DATA files. This copying is done automatically in the GUIDE EXEC file, which is described below.

The CMS system was carefully designed for tailoring by system programmers to users' specifications. Consequently, CMS features separate file definition commands with a large number of parameters. In addition, CMS has a facility (EXEC file) for easily condensing large quantities of job control instructions into single-line commands.

A number of useful EXEC files on the SPAHR disk are designed to simplify alteration and maintenance of the SPAHR code. Each of these files is self-documenting: When the user enters a command of the form filename ?

the EXEC responds with a description of what that file does and how it is used. Because the specifications for these EXEC files may change with time, they will not be described in detail here. Only a brief indication of their purpose will be given.

SPAHR	is the EXEC file for executing the SPAHR module file. It is described in detail in Chapter 2 of Volume IV, the SPAHR User's Guide, and in Chapter 2.1 of Volume II, the SPAHR Introductory Guide.
DEMTAPE	creates and submits a job to the OS system, which creates a SPAHR distribution tape.
DEMLOAD	creates and submits jobs to the OS system to create and manage the OS version of the SPAHR program at ANL.
FTN	compiles, loads, and executes the experimental code modifications using the FORTRAN H extended, enhanced compiler.
WATV	compiles, loads, and executes the experimental code using the WATFIV debugging compiler.
LIBGN	breaks the experimental code into individual subroutine files, which are then used to replace the equivalently named FORTRAN, TEXT (i.e. object code), and MACLIB members in the D disk. LIBGN should only be used when the altered routines have been debugged and are ready for permanent installation.
WATLIB	takes the FORTRAN files and creates a new MACLIB. WATLIB is needed only if the MACLIB has been damaged and must be totally recreated.
USE	summons forth the WYLBUR text editor on the indicated file.
GUIDE	creates selected SCRIPT files and generates a current copy of the SPAHR documentation.

BIBLIOGRAPHY

Carnow, B. W. and P. Meier 1973: "Air Pollution and Pulmonary Cancer," Arch. Env. Health. 27:207-218.

Chiang, C. L. 1968: Introduction to Stochastic Processes in Biostatistics, Wiley and Sons, New York.

Comar, C. and L. Sagan 1976: "Health Effects of Energy Production and Conversion," in Hollander, J. M., Ed., Annual Review of Energy, Vol I:581-600, Annual Reviews, Inc., Palo Alto.

Finch, S. J. and S. C. Morris 1977: Consistency of Reported Health Effects of Air Pollution, BNL-21808, Brookhaven National Laboratory, Upton, NY.

Hammel, E. A., D. W. Hutchinson, K. W. Wachter, R. T. Lundy, and R. Z. Deuel 1976: The SOCSIM Demographic-Sociological Microsimulation Program, Institute of International Studies Monograph No. 27, Institute of International Studies, Berkeley.

Hammond, E. C. 1966: "Smoking in Relation to the Death Rates of One Million Men and Women," in Haenzel, W., Ed., Epidemiological Study of Cancer and Other Chronic Diseases, National Cancer Institute Monograph 19, U. S. Public Health Service, Washington, D. C.

Keyfitz, N. 1968: Introduction to the Mathematics of Population, Addison-Wesley, New York.

Keyfitz, N. and W. Flieger 1968: World Population: An Analysis of Vital Data, University of Chicago Press, Chicago.

Keyfitz, N. and W. Flieger 1971: Population: Facts and Methods of Demography, W. H. Freeman, San Francisco.

Lave, L. and E. P. Seskin 1977: Air Pollution and Human Health, Johns Hopkins University Press, Baltimore.

Lindberg, W. 1968: The General Air Pollution in Norway, Norwegian Smoke Damage Council, Oslo.

Lundy, R. T. and D. Grahn 1977: "Predictions of the Effects of Energy Production on Human Health," Proceedings of the Social Statistics Section, Part II: 672-675, American Statistical Association, Chicago.

Morris, S. C. and K. M. Novak 1976: Handbook for the Quantitation of Health Effects due to Coal (Draft), Brookhaven National Laboratory, Upton, NY.

National Academy of Sciences, Advisory Committee on the Effects of Ionizing Radiation 1972: The Effects on Populations of Exposure to Low Levels of Ionizing Radiation, National Academy of Sciences, Washington, D. C.

National Academy of Sciences, Advisory Committee on the Effects on Ionizing Radiation 1980: The Effects on Populations of Exposure to Low Levels of Ionizing Radiation, National Academy of Sciences, Washington, D. C.

Preston, S. H., N. Keyfitz, and R. Schoen 1972: Causes of Death: Life Tables for National Populations, Seminar Press, New York.

Schoen, R. 1970: "The Geometric Mean of the Age-Specific Death Rates as a Summary Index of Mortality," Demography 7(3):317-324.

Thomas, T. 1973: An Investigation into Excess Mortality and Morbidity due to Air Pollution, Ph.D. Thesis (Environmental Science), Purdue University.

United Nations, Department of Economic and Social Affairs, Population Branch 1967: Methods of Estimating Basic Demographic Measures from Incomplete Data, Manual IV, Population Study No. 42, United Nations, New York.

U. S. Nuclear Regulatory Commission 1975: Reactor Safety Study: Appendix VI Calculation of Reactor Accident Consequences, WASH-1400 (NUREG 75/014) USNRC, Washington, D. C.

U. S. Nuclear Regulatory Commission 1976: Final Generic Environmental Statement on the Use of Recycled Plutonium in Mixed Oxide Fuel in Light Water Cooled Reactors, Vol. III, Chapter IV, NUREG-0002, Washington, D. C.

Wilson, R. 1972: "Tax the Integrated Pollution Exposure," Science 178:182-183.

Winkelstein, W. Jr., S. Kantor, E. W. Davis, C. S. Maneri, and W. E. Mosher 1968: "The Relationship of Air Pollution and Economic Status to Total Mortality and Selected Respiratory System Mortality in Men II: Oxides of Sulfur," Arch. Env. Health 16:401-405.

Distribution for NUREG/CR-2364 (ANL-81-59)

Internal:

M. K. Butler
D. Grahn
M. W. Rosenthal (4)
F. S. Williamson
ANL Patent Dept.
ANL Contract File
ANL Libraries (2)
TIS Files (6)

External:

USNRC, for distribution per RH (205)
DOE-TIC (2)
Manager, Chicago Operations Office, DOE
President, Argonne Universities Association
Division of Biological and Medical Research Review Committee:
 T. W. Clarkson, U. Rochester
 L. Grossman, Johns Hopkins U.
 J. W. Osborne, U. Iowa
 H. C. Pitot III, U. Wisconsin-Madison
 M. Pollard, U. Notre Dame
 F. W. Putnam, Indiana U.
 P. O. P. Ts'o, Johns Hopkins U.
U. S. Nuclear Regulatory Commission:
 E. Branagan
 J. D. Foulke
 M. E. Ginevan
 R. Gotchy
 C. Willis
U. S. Department of Energy:
 N. F. Barr
 C. W. Edington
 R. Goldsmith
 M. L. Minthorn, Jr.
 J. W. Thiessen
J. J. Collins, American Cyanamid Co., 1 Cyanamid Plaza, Wayne, NJ 07470 (21)
C. Land, National Cancer Institute, Bethesda, MD 20205
R. T. Lundy, Alameda, CA 94501